Programmable Infrastructures for Secure Healthcare

Jamila Alsayed Kassem

This thesis tackles the pressing challenge of secure and efficient health data sharing, proposing the Enabling Personalized Intervention (EPI) framework designed to facilitate the data collaboration process across healthcare institutions.

This thesis introduces a framework that provides seamless, policy-compliant sharing capabilities within a multi-party environment. The EPI framework is structured to support a flexible yet highly secure infrastructure, accommodating the diverse needs of healthcare providers who must share data across institution boundaries.

This thesis is a collection of Jamila's work between 2019 and 2023 under the supervision of Prof. dr. P. Grosso, Prof. dr. ir. C.T.A.M. de Laat, and Prof. dr. A. Osseyran.

Programmable Infrastructures for Secure Healthcare

Jamila Alsayed Kassem

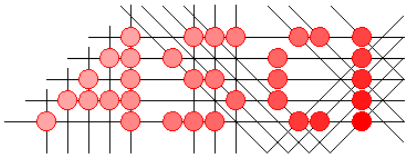
Programmable Infrastructures for Secure Healthcare

Jamila Alsayed Kassem

Programmable Infrastructures for Secure Healthcare

Jamila Alsayed Kassem

The EPI project is funded by the Dutch Science Foundation in the Commit2Data program (grant no: 628.011.028).



The study was completed at ASCI, and the dissertation number is 466.

Cover design by Jamila Alsayed Kassem using www.midjourney.com.
Printed and bound by Proefschriftspecialist.nl – Zaandam.

ISBN: 978-94-93391-81-9

Programmable Infrastructures for Secure Healthcare

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in de Agnietenkapel

door

Jamila Alsayed Kassem

geboren te Tamnine Eltahta

Promotiecommissie

Promotor:	Prof. dr. ir. C.T.A.M. de Laat	Universiteit van Amsterdam
	Prof. dr. P. Grosso	Universiteit van Amsterdam
Co-promotor:	Prof. dr. A. Osseyran	Universiteit van Amsterdam
Overige leden:	Prof. dr. C.I. Sanchez Gutiarrez	Universiteit van Amsterdam
	Prof. dr. S. Klous	Universiteit van Amsterdam, KPMG
	Prof. dr.ir. H.E. Bal	Vrije Universiteit Amsterdam
	Dr. A.M. Oprescu	Universiteit van Amsterdam
	Dr. F. Regazzoni	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Contents

Chapter 1

Introduction

In the scope of medical research, secure data sharing exploiting the programmability of digital infrastructures for healthcare is a powerful catalyst for progress. That is evident in the substantial advancements achieved over the last century in preventing, curing, and treating diverse conditions. Notable examples include the established correlations between smoking and cardiovascular disease (13) and cancer (88), along with the identified link between low B vitamin during pregnancy and the heightened risk of neural tube defects (17). While these instances underscore the impact of data analysis on healthcare, they represent just a fraction of its broader contributions to improving the body of medical research.

The increasing volume of health-related data necessitates robust and secure data-sharing mechanisms, crucial for unlocking its full potential in advancing personalized medicine and the development of Digital Health Twins (DHT). Aggregating diverse patient data, electronic health records, genomics, medical imaging, wearable devices, and lifestyle information lays the foundation for innovative data-centric health applications.

Personalized medicine emerges as a transformative paradigm within healthcare, characterized by its focus on tailoring medical treatments and interventions to the individual characteristics of each patient. This approach recognizes that variations in genetics, environment, and lifestyle can significantly influence an individual's response to treatment, highlighting the importance of a more targeted and precise approach to healthcare delivery.

Moreover, larger sample sizes not only enable the breakdown of population heterogeneity into more uniform groups but also hold promise in enhancing the specificity of understanding the genesis of specific traits or illnesses. This capacity extends to the development of more individualized models, with larger sample sizes providing the opportunity to apply sophisticated machine-learning models to the data, hence moving closer toward personalized medicine.

On the other hand, the lack of data sharing in healthcare significantly hinders medical research, slows the development of new treatments, and reduces the

quality of patient care by limiting access to comprehensive information. This inefficiency leads to duplicated efforts, increased costs, and potential misdiagnoses. It also suppresses innovation in technologies like AI, exacerbates healthcare inequalities, and restricts public health responses to epidemics. Legal, ethical, and privacy concerns further complicate the situation, making it essential to develop standardized protocols and robust security measures to facilitate data sharing while protecting patient privacy. By harnessing the power of data sharing and advanced analytics, personalized medicine holds the potential to revolutionize healthcare delivery, improving patient outcomes and driving greater efficiencies across the healthcare systems.

1.1 Digital Health Twins

The concept of Digital Twins (DTs) is not new, and it originally appeared in the early 1990s (42) under different terminologies, such as the "Mirror Space Model" (45), "Information Mirror Mode"(46), etc. Over the past decade, the idea has been gaining more and more attention from researchers and industry. Although DTs have been primarily discussed within engineering and industrial contexts, medical and health use cases are not excluded from the DTs' impact.

In theory, a DT is a digital copy of a physical single object, service, or complex system. In practice, a DT is a digitized model that dynamically couples both virtual and physical twins, and makes use of contemporary technologies like smart sensors (IoT devices) and data analytics to predict and identify failures, discover and simulate optimizing opportunities, and improve outcomes (60). A Digital Health Twin (DHT) can potentially mirror the human body itself (or a part of it) Subsequently, there exist different types and use cases of DHT that represent different aggregates of data and algorithms. As an example, we model a single bodily function by creating a DHT of the digestive system, or model a healthcare institution DHT, to monitor, optimize, and provide interventions throughout the treatment cycle. This is pivotal to facilitate medical research such as drug composition, patient rehabilitation, and treatment plans.

Deploying a DHT utilizes medical data-sharing and patient-generated data to empower personalized medicine. Personalized medicine is a novel approach to improving clinical care using an individualized or stratified approach to diagnosis and treatment rather than a group treatment approach (83). With that in mind, DHT comes as a natural, complementary approach to implementing personalized medicine, since it offers the capacity to model a distinct patient with varying physiological features and genetic differences.

In the context of personalized medicine, data collaboration among healthcare providers is vital for effective medical data and Electronic Health Records (EHR) sharing. The sensitive nature of this information emphasizes the need for secure transmission, storage, and processing. This presents a challenge in

dynamically establishing a collaborative data-sharing environment across diverse healthcare domains. A comprehensive data-sharing framework is necessary to unite the efforts of research centres, health institutions, and patient groups, considering data-sharing policies, infrastructural capabilities, and dynamic matching of security requirements.

Moreover, recent events emphasize the ethical use of data subjects, promoting a culture of data sharing while safeguarding individuals' privacy via policies and laws, like the GDPR([26]) and HIPAA([6]). Balancing data protection and sharing practices varies among institutions and countries, requiring researchers to operate within legal boundaries. Efficient and secure data sharing among healthcare providers is pivotal for utilizing the power of vast and diverse data, enabling advanced machine learning algorithms, artificial intelligence models, predictive analytics, precision medicine, and the creation of Digital Health Twins. Note that the term "data sharing" also includes algorithms and derived knowledge secure sharing.

The variability of said policies further reinforces the need for an adaptive framework to dynamically set up this policy-abiding collaborative environment. On top of that, more challenges arise when considering the framework's design requirements, such as reliability requirements, network performance requirements, set-up overhead, and resource limitations. The adoption of DHTs is accompanied and accelerated by maturing and growing computing technologies. This is led by cloud computing and network virtualization, which offer the means to facilitate knowledge discovery by provisioning on-demand computing and network resources. One of the main contributing factors toward the successful and reliable deployment of DHTs is the underlying network paradigm connecting all the data-sharing components to effectively run a use case ([82]).

1.2 Dynamic Infrastructures

Given the problems above, secure data sharing among healthcare providers is still unresolved. Programmable and virtualized infrastructure can provide the mechanisms to support this. In fact, the present generation of ICT infrastructures heavily relies on virtualization, given the successful evolution of virtualisation over the past decades ([49]). The Network Function Virtualisation (NFV) architecture, standardized by the European Telecommunications Standards Institute (ETSI), serves as a foundation for defining the subsequent generation of network infrastructures ([1]). This architecture is designed to manage and coordinate network resources for cloud-based applications and the lifecycle of network services. Furthermore, the NFV paradigm enables the on-demand implementation and instantiating of Network Functions (NFs) like firewalls, segmentation, and Deep Packet Inspection (DPI), which is especially crucial in managing diverse collaborative domains.

The emergence of Software-Defined Infrastructures (SDIs) introduces programmable infrastructures managed by centralized control plane components. SDI integrates the control and management of varied computing and networking resources through software, providing ample programmability. This programmability can be leveraged to develop infrastructure-independent NFV frameworks, enhancing cross-domain innovation through open interfaces. These complementary technologies offer the potential to construct a dynamic network infrastructure capable of adapting to diverse healthcare application requests and requirements.

Figure 1.1 illustrates the framework's general overview developed in the EPI Enabling Personalised Interventions) project, where the end users interact with the framework to apply simple to complex workflows. The framework should adapt to the plethora of requirements, including security considerations, to run the requested workflow on different types of medical data.

1.3 Research Questions

To tackle the challenges of deploying personalized medicine use cases and medical data sharing, we formulate the **main research question** of this manuscript:

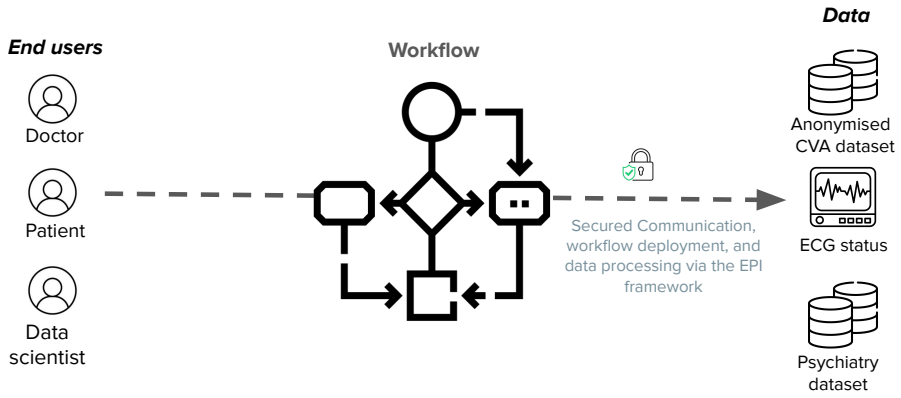
RQ: How can we effectively support health data sharing use cases for heterogeneous parties collaborating within a low-level programmable virtualizable digital infrastructure?

To answer this question, we delve into a number of sub-questions starting with first identifying the scope of the project and hindering challenges. The potential research advancement with medical data sharing has been discussed in the current body of literature. However, data privacy protection and ethical considerations that are involved tend to discourage a lot of these efforts, by enforcing data protection laws and requirements. Chapter 2 introduces the Enabling Personalised Intervention (EPI) project that aims to advance personalised medicine research efforts by addressing these challenges on multiple levels. Moreover, we introduce the data-sharing use cases that are at the centre of our studies. These use cases serve as exemplary applications, on our road toward DHT deployment.

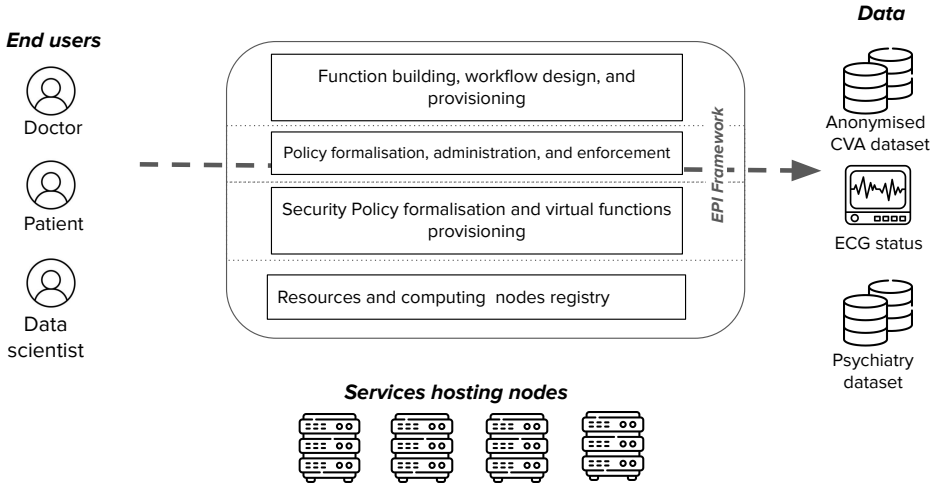
Chapter 2 primarily focuses on the DHT use cases, data-level challenges, and study models. This will illustrate the type of data we are dealing with, and the data flow expected. We build upon this information in Chapter 3 to discuss the EPI Framework design considerations. This leads to the following sub-question:

- RQ1: "How can we address open challenges in the context of policy, security, and computing data sharing via building a dynamic infrastructure framework to deploy DHT use cases?"

In Chapter 3 we partially answer this question by introducing the EPI framework,



(a) The EPI framework is utilised to secure all communication, workflow deployment, data sharing and processing.



(b) The EPI framework has four main components: 1) workflow provisioning feature, 2) policy reasoning feature, 2) security provisioning feature, 3) and lastly, the available resources and computing nodes that are available for hosting services.

Figure 1.1: This figure illustrates the high-level concept of the dynamic EPI framework; where on the left side there are the end users that will be utilising the framework and submitting the workflow they want to run to the EPI framework. As a result, on the right, the EPI framework should be able to run the workflow on different types of medical data that are required for successful processing. This needs to be done securely.

an adaptive medical data-sharing framework. We define the different components of the EPI framework; including the policy reasoner, the workflow orchestrator, and the infrastructure orchestrator. The policy reasoner introduces automated policy interpretation, management, and enforcement. This allows us to formulate a wide range of data-sharing policies and translate that into setup actions that could be considered by the framework. Furthermore, the workflow orchestrator (the BRANE orchestrator) provides the means to formulate workflows to run the use cases via mapping the request to the available computing, storage, and network resources and orchestrating the workflow after checking and planning with the policy reasoner. Lastly, at a network level, the network and security requirements are enforced via instantiating on-the-fly services. This is done in compliance with the policy and planned workflow deployment. Chapter 3 serves as an overview of the EPI Framework, and discusses the proof of concept deployed as part of the EPI project efforts. We further elaborate on the different features of the framework in later chapters; namely, the distinction between high-level policies and low-level capabilities.

In Chapter 4, we make a distinction between high-level policies dictating allowed data flows and low-level available infrastructure requirements. This distinction is made to map what is ideally required by the policies, and what is possible while setting up the collaborative data-sharing infrastructure. In this chapter, we consider "secure" data flows as permissible flows. We address the challenge of automatically managing these policies, and adapting the underlying infrastructure accordingly. We formulate the security area logic model to automate policy aggregation with feasible routes. We evaluate the framework's performance by measuring setup overhead with highly sparse security area distributions, which will indicate more setup actions needed. The EPI framework is based on the main design feature which is the capability of managing data traffic and redirecting data flow routes. This brings us to the following sub-question:

- RQ2: "What are the performance tradeoffs when deploying different packets' redirection methods and virtual network functions to enforce network policy-compliant routes under different workloads?"

A major design consideration of the EPI Framework is the timely setup of the collaborative infrastructures. Some DHT use cases could be time-sensitive, and minimising network overhead is one way of addressing this. For example, if the use case requires real-time patient status monitoring. Chapter 5 focuses on the traffic redirection feature implementation employing SOCKS-based and NGINX-based proxies. This empowers policies since the EPI Framework can enforce that by setting up extra security and bridging functions (such as access control portals) and re-routing traffic via those functions. We analyse the network traffic with different proxies to pinpoint the source of delays with each one. We base the recommendation for utilising the proxy on the benchmarking study we have.

Other than the proxy, the extra security and bridging functions instantiated could be the point of failure affecting the EPI framework's reliability and performance.

In Chapter 6, we implement several Bridging Functions Chains (BFCs), and this serves as the different security and network functions we will be deploying with our use cases. We, then, build profiles of resource consumption of different combinations of chains. We consider high-load use cases (like data streaming) and lower-load use cases, in an effort to stress-test the infrastructure setup. After building these profiles, we gain some insights into the different BFCs' resource requirements, and then we ask:

- RQ3: "How can we automate an adaptive Service Function Chain Provisioning on available network Points of Placement candidates to run a data-sharing request under different use cases' requirements?"

The previous chapters look into formulating policies, aggregating policies, and enforcing them by adaptable re-routing traffic via BFCs. In Chapter 7, we start looking into efficient methods for BFC provisioning. We introduce the problem statement of the BFC provisioning and conclude that it is a constrained search problem, such that we are searching for the optimal placement of BFC while considering multiple constraints. These constraints are namely resource availability, time-out, and chainability (whether a path exists between functions in a chain) limitations. We employ different provisioning methods: greedy heuristic, Deep Q-learning (DQL), and heuristic-boosted DQL placement methods. We evaluate and compare these methods and build our recommendation accordingly.

Lastly, we provide a holistic overview of the EPI framework, and we evaluate whether it can provide secure and private medical data sharing, so we ask ourselves:

- RQ4: "How can we orchestrate the EPI framework services according to *privacy-by-design* principles by comprehensively modelling privacy probabilities and mitigating risks of DHT data-sharing workflows according to privacy-defined attributes?"

In Chapter 8, we discuss again the EPI Framework in the context of *privacy-by-design* considerations. We define the privacy risk assessment model to quantify privacy risk and highlight the benefits of utilising our framework in minimising privacy risk. We list important privacy attributes, and we also address the effect of maximising data privacy on data utilisation (data quality).

1.4 Key Contributions

- In Chapter 3, we design and implement a Kubernetes-based data-sharing framework that enables running distributed workflows securely and privately, compliant with specific policies.

- In Chapter 4, we propose a methodology to automatically aggregate allowed data flow with feasible data flow based on the infrastructure resources and policies within the EPI use cases.
- In Chapters 5 and 6, we implement the EPI traffic redirection and function chaining features, and we provide benchmarks and performance profiles.
- In chapter 7, we build adaptive BFC provisioning techniques based on Heuristic, deep Q-learning, and heuristic-boosted deep Q-learning deployed running the EPI use cases.
- In chapter 8, we provide a new type of risk model that is based on security and privacy considerations and relates that to relevant threats, data utility, and mitigation techniques.

1.5 Publications

Listed below are the published and submitted papers that were used in the chapters, in addition to the author's contributions.

- J. A. Kassem, C. Allaart, S. Amiri, M. Kebede, T. Müller, R. Turner, A. Belloum, L. T. v. Binsbergen, P. Grunwald, A. v. Halteren, P. Grosso, C. d. Laat, and S. Klous. Building a Digital Health Twin for Personalized Intervention: The EPI Project. In Commit2Data. Open Access Series in Informatics (OASIcs), Volume 124, pp. 2:1-2:18, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2024) https://doi.org/10.4230/OASIcs.Commit2Data.2

  J.A.K. wrote the distributed analysis infrastructure subsection, organised the writing, consolidated the work and did the final editing. C.A. wrote the subsection relating to the Personalized predictions for stroke rehabilitation use cases. S.A. wrote the subsection relating to the distributed data analysis. M.K. wrote the policy formalisation and management part. T.M. wrote the subsection relating to the workflow orchestrator and the EPI proof of concept. R.T. wrote the real-time response for the Psychiatry use case part. The remaining co-authors edited and supervised the written work.

- J. A. Kassem, C. de Laat, A. Taal and P. Grosso, "The EPI Framework: A Dynamic Data Sharing Framework for Healthcare Use Cases," in IEEE Access, vol. 8, pp. 179909-179920, 2020, doi: 10.1109/ACCESS.2020.3028051.

  J.A.K. worked on conceptualizing and designing the EPI infrastructure orchestrator, worked on the methodologies and developed the area logic model. J.A.K. ran the experiments and the evaluation method. The remaining co-authors consulted the study, and supervised, and edited the written work.

- J. A. Kassem, O. Valkering, A. Belloum and P. Grosso, "EPI Framework: Approach for Traffic Redirection Through Containerised Network Functions," 2021 IEEE 17th International Conference on eScience (eScience), Innsbruck, Austria, 2021, pp. 80-89, doi: 10.1109/eScience51609.2021.00018.

  J.A.K. and O.V. worked on implementing the redirection proxies and the benchmark experiments. J.A.K. and O.V. ran the experiments and provided data analysis and evaluation. The remaining co-authors consulted the study, edited the work, and supervised.

- J. A. Kassem, A. Belloum, T. Müller and P. Grosso, "Utilisation Profiles of Bridging Function Chain for Healthcare Use Cases," 2022 IEEE 18th International Conference on e-Science (e-Science), Salt Lake City, UT, USA, 2022, pp. 475-480, doi: 10.1109/eScience55777.2022.00085.

  J.A.K. worked on developing the Bridging functions, the Kubernetes environment, and the chaining of functions. J.A.K. ran the experiments, built the profiles, and provided data analysis. The remaining co-authors consulted, edited, and supervised the work.

- J. A. Kassem, L. Zhong, A. Taal and P. Grosso, "Adaptive Services Function Chain Orchestration For Digital Health Twin Use Cases: Heuristic-boosted Q-Learning Approach," 2023 IEEE 9th International Conference on Network Softwarization (NetSoft), Madrid, Spain, 2023, pp. 187-191, doi: 10.1109/NetSoft57336.2023.10175506.

  J.A.K. conceptualised the service function chaining problem statement and the study methodology. J.A.K. and L.Z. worked on the algorithm implementations. J.A.K. ran the experiments, provided data analysis, and consolidated the work. The remaining co-authors consulted the study, edited the work, and supervised.

- J. A. Kassem, T. Müller, C. A. Esterhuyse, M. G. Kebede, C. de Laat, A. Osseyran and P. Grosso, The EPI framework: A data privacy by design framework to support healthcare use cases, Future Generation Computer Systems, 2024, 107550, ISSN 0167-739X, https://doi.org/10.1016/j.future.2024.107550.

  J.A.K. worked on the privacy by design study, in addition to the privacy risk model formalisation. J.A.K. worked on the infrastructure orchestrator design, the walk-through, and the evaluation subsection. J.A.K. consolidated the work and did the final editing. T.M. worked on the workflow orchestrator design. C.A.E. and M.G.K. worked on the policy part. The remaining co-authors edited and supervised the work.

1.5.1 Code repositories

- https://github.com/epi-project/brane
- https://github.com/epi-project/proxy-bench
- https://github.com/epi-project/Netsoft2023

1.6 Thesis Overview

Figure 1.2 visualizes the thesis organisation. This thesis contains 9 chapters, including the introduction and conclusion. The figure depicts the major points discussed in each chapter, and the research questions addressed.

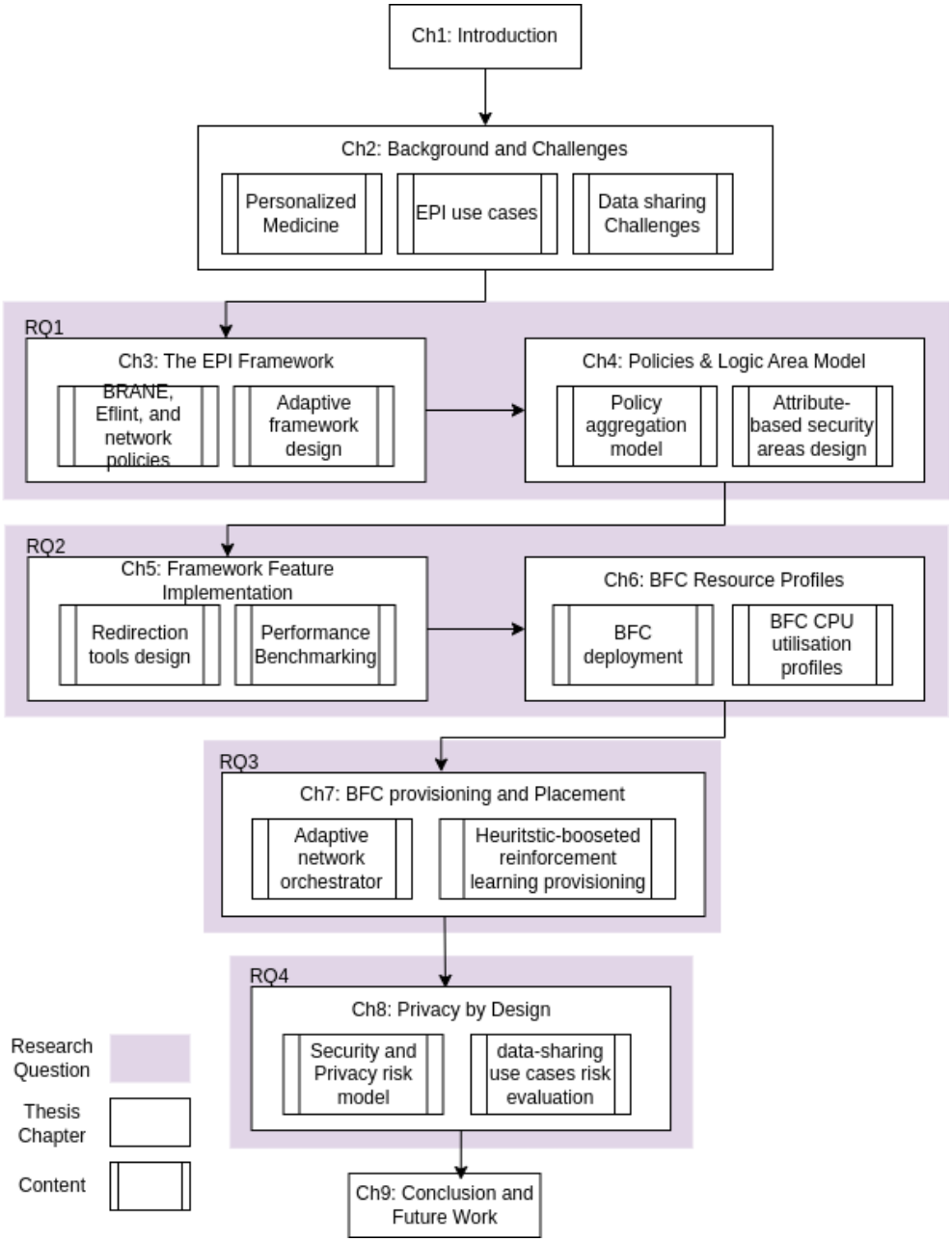


Figure 1.2: The thesis chapter overview including chapters' organisation, key points, and research questions.

Chapter 2

The Enabling Personalized Intervention (EPI) Project: An Introduction

The Enabling Personalized Interventions (EPI) project ([2]), which ran from 2019-2024, brings together research on data science, software-defined network infrastructure, and secure data sharing, deployed within the healthcare domain. The project explores the digital twin paradigm, in which data science-driven algorithms monitor and perform functions on a digital counterpart of a real-world entity, to enable proactive responses based on predicted outcomes. The EPI project applies this paradigm in the healthcare context by developing and testing applications that can act as personalized digital health twins for self/ joint healthcare management.

This chapter gives an introduction to the EPI project, and we report on the use cases the project references as a basis to develop possible solutions to the data-centric advancement challenges. In particular, we describe algorithms and tools for algorithmic real-time response and analysis of distributed data at scale.

This chapter is based on:

- **Jamila Alsayed Kassem**, Corinne Allaart, Saba Amiri, Milen Kebede, Tim Müller, Rosanne Turner, Adam Belloum, L. Thomas van Binsbergen, Peter Grunwald, Aart van Halteren, Paola Grosso, Cees de Laat, and Sander Klous. Building a Digital Health Twin for Personalized Intervention: The EPI Project. In Commit2Data. Open Access Series in Informatics (OASIcs), Volume 124, pp. 2:1-2:18, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2024) https://doi.org/10.4230/OASIcs.Commit2Data.2

2.1 Introduction

Recent breakthroughs in healthcare have been greatly influenced by improved technologies that enable early diagnosis. Data analytic technologies have strong potential to help solve complex problems in many industries: knowledge is power – and in healthcare, that holds in particular. Yet, for an industry that is under financial stress, the increasing complexity of disease and comorbidity, data-centric advances in health have been burdened by capability constraints – why has data analysis not been healthcare's saviour? Several challenges have inhibited this. For example, data is not accessible and remains (predominantly) in silos; data is not widely analysed to derive meaningful clinical insights; insights are not accessible as actionable information for healthcare providers or patients to self/joint manage the patient's condition. The project described in this publication - EPI - Enabling Personalized Intervention[1] - addresses these challenges with the ultimate objective of improving cost efficiency, quality, and outcomes of care, while ensuring patient and public health data and results are processed safely and with respect for the digital (privacy) rights of patients.

The EPI project vision is structured around the concept of the *digital health twin* (DHT) as defined by (19). "Digital Twins stand for a specific engineering paradigm, where individual physical artefacts are paired with digital models that dynamically reflect the status of those artefacts." This leads to the hypothesis that with a DHT: "one would be in the possession of very detailed bio-physical and lifestyle information of a person over time. This perspective redefines the concept of 'normality' or 'health' as a set of regular patterns for a particular individual, against the backdrop patterns observed in the population." The development of personalized digital health twin algorithms will inherently benefit from the ability to collect as much relevant data as possible to cluster patient groups better and cater diagnosis treatment and outcome prediction according to the patient's genetic makeup and medical history. Given the emphasis on the individual and the dynamic nature of a DHT, our main research question is if the existence of a DHT indeed enables instant (real-time), effective, personalized guidance to prevent health-related incidents and/or helps improve intervention effectiveness.

The paper starts by going over the project organization in Section 2.2, and then we introduce the DHT use cases investigated in Section 2.3. We address the challenges of building a real-time algorithmic response to enable collecting and analysing patients' data within a DHT use case in Section 2.4. On the other hand, data is often distributed and hosted across healthcare domains, and we discuss the methods to analyse data at scale utilizing distributed learning algorithms and privacy-preserving machine learning in Section 2.5.

[1] https://enablingpersonalizedinterventions.nl/

2.2 Project organization: partners roles and partners interactions

The EPI project was funded by the Commit2Data Data2Person call from NWO, the Dutch national funding agency. The project partners started their work in 2019 and concluded their collaboration in 2024.

Already in the preparation phase of the project, it was clear to all involved that the only chance to develop a working solution was to create a cooperation of an interdisciplinary team of medical specialists, data scientists, ICT infrastructure experts as well as experts in artificial intelligence and law. This is what in the end occurred: in fact, the EPI project brings together academic and research partners, healthcare providers and the private sector to tackle the challenges related to enabling personalized intervention.

The University of Amsterdam (UvA), the Centrum Wiskunde and Informatica (CWI) and the Vrije University Amsterdam (VUA) provide their scientific expertise and employ PhD researchers to work closely with the other project partners. The three hospitals within the consortium, St. Antonius, Princess Maxima Center, and University Medical Center Utrecht (UMCU) provide three representative health-related case studies. The use cases all have their requirements for the operational analysis environment. It is not scalable nor desirable to have to manage yet another environment for each use case: the infrastructure developed as part of EPI solves that problem. To accomplish this, the medical institutions have put their domain experts in contact with the researchers at the universities and created close collaborations on the individual use cases. SURF, also involved in the project, provides the infrastructural components; it fulfils the role of platform manager, established to manage the operation of the platform and to monitor and enforce that its participants behave as expected. Finally, KPMG and Philips as commercial partners in the project contribute insightful direction towards the long-term sustainability of the EPI results. During the whole duration of the project, all partners have maintained very close interactions, even in periods of Covid lockdowns.

As the project is now in its concluding phase, the partners are actively working for the results to see broader adoption, beyond the time span of the project. EPI offers models, tools, and software that are also broadly applicable, not just to the medical field. A first and concrete outcome of these efforts is the collaboration of EPI with the Amsterdam Data Exchange (AMdEX) project [2]. AMdEX is an innovation field lab initiated by AMS-IX, SURF, UvA, DEXES and the Amsterdam Economic Board and co-funded by the European Regional Development Fund. The EPI partners have already acquired funding for integration of the EPI result in AMdEX, and are looking for further commercialization possibilities.

[2]This work is partially funded through the AMdEX Fieldlab project supported by Kansen Voor West EFRO (KVW00309) and the province of Noord-Holland.https://amdex.eu

2.3 Use Cases

2.3.1 Personalized predictions for stroke rehabilitation (St. Antonius Hospital)

The goal of this use case is to provide personalized outcome predictions for the rehabilitation of patients who suffered from a cerebrovascular accident (CVA), utilizing the data of these patients that is distributed among different care institutions, in a privacy-preserving manner.

Outcome predictions based on the complete care path:

Following acute care for a cerebrovascular accident (CVA) in the hospital, patients commonly undergo therapy through either inpatient or outpatient rehabilitation. Given the diverse conditions and life stages of CVA patients, rehabilitation treatments and long-term health outcomes post-rehabilitation vary significantly (77). Despite the shared belief in the importance of personalized prognoses among healthcare professionals, limited knowledge exists regarding these outcomes upon hospital discharge.

To address the need for personalized outcome predictions and to establish an overview of the care continuum, this project aims to develop a DHT encompassing the entire care trajectory of CVA patients. However, this task is complex due to the involvement of multiple healthcare institutions—hospitals, rehabilitation clinics, and nursing homes—each housing a portion of the patient's data. Organizational, compatibility, and privacy concerns frequently arise in this scenario. The objective is to generate personalized outcome predictions while safeguarding data privacy through a distributed approach that constructs prediction models while preserving the privacy of data from participating clinics.

Clinical interface to assist with shared decision-making:

In the healthcare sector, there has been a movement towards Shared Decision-Making (SDM) (33)), whereby healthcare professionals devise a care plan not solely based on clinical expertise but also by actively considering the personal preferences and needs of each patient. For individuals who have suffered a cerebrovascular accident (CVA), a diverse and extensive patient population, post-acute care rehabilitation presents an opportunity where SDM can prove advantageous (74).

To facilitate the SDM process, our objective is to develop a clinical interface that offers personalized insights into patient prognosis based on various care

paths. This interface will be founded on distributed prediction models, delineating the anticipated outcomes of distinct individual care trajectories to assist clinicians in the SDM process.

2.3.2 Personalized recommendations in Psychiatry

The goal is to provide personal and explainable treatment recommendations to accelerate the process of finding the right treatment for individual psychiatry patients.

Determining the information that the clinical interface should display:

Our objective is to design a clinical interface where patients and their clinicians, based on their DHT records from the psychiatry department, receive insightful treatment recommendations and prognostications (Section 2.4). However, determining the most beneficial nature of these recommendations for patients and clinicians remains unclear. Within psychiatry, there exists no consensus regarding the optimal methods for assessing and forecasting treatment effects, particularly for diverse patient populations with multiple comorbidities. Presently, clinical trials utilize numerous symptom rating scales, which may not be suitable for heterogeneous patient groups, alongside quality of life assessments and measures of autonomy, social engagement, and side effects (102).

To address this issue, interviews were conducted with patients and clinicians at participating mental health hubs to identify the most valuable treatment outcomes for prediction and presentation. Additionally, the feasibility of extracting proposed outcome measures from the digital health twin was explored, including the extraction of information from both structured and unstructured data. Moreover, the potential for real-time extraction, wherein new patient outcome data, such as reported side effects during clinical visits, could be transferred to the DHT records and utilized by the recommender system to adjust predictions, was investigated. Detailed findings of this study can be found in (102).

Developing personalized recommender systems for psychiatry patients:

Psychiatric syndromes exhibit considerable heterogeneity, and the occurrence of side effects significantly influences treatment adherence and outcomes. Thus, tailoring the predicted outcomes of the recommender algorithm could substantially enhance the efficiency of finding the appropriate treatment. One patient may prioritize minimizing side effects, while another may prioritize a swift return to work. In a third scenario, a clinician may identify a convergence of two syndromes, where understanding the interaction between two symptoms is pivotal in the progression of the disease, prompting early targeting of these specific

symptoms during treatment.

To facilitate this customization, our objective is to develop and train a recommender algorithm capable of predicting the most suitable treatment for individual patients, considering either a single outcome or a combination of outcome measures. Within the clinical user interface, both the clinician and patient will collaboratively determine the weighted combination before receiving recommendations.

Sharing recommender systems between mental health hubs:

Once a recommender system has undergone training and validation, it will be shared with other mental health hubs for two primary purposes. Firstly, it is imperative to demonstrate the generalizability of medical recommender systems by validating them in independent patient cohorts before their implementation in clinical practice. Secondly, post-validation, the algorithm should be shared to enhance the records across various mental health hubs.

To facilitate the exchange of the algorithm, a secure infrastructure must be established to enforce the policies of all participating mental health hubs. This infrastructure should regulate access to different components of the recommender system, ensuring that individual patient data cannot be accessed through requests to the shared algorithm. Additionally, the sharing platform should integrate the diverse informed consent systems utilized by each mental health hub.

A significant challenge, not currently addressed in this project, is the potential disparity in the structure of data employed by each hub. This discrepancy becomes problematic when the recommender system processes structured data as input or output. For instance, one hub may store a patient's previous medication in a structured format managed by the pharmacist, while another hub may record previous medication as free text in the electronic health records documented by clinicians. As a result, when our system is implemented in the first hub, it may struggle to locate the relevant information in the second hub. Preprocessing the data of digital health twins at each hub may necessitate automation for future real-time applications, possibly through the introduction of a meta-layer surrounding the recommender algorithm.

2.3.3 Princess Maxima Center

The EPI project's goal for this use case is to enable compliant data-sharing by automating privacy policies from privacy legislation and contractual agreements on data-sharing to facilitate stakeholder collaboration in discovering new treatments and prognosis factors for the Diffuse Intrinsic Pontine Gliomas (DIPG) disease, a rare and deadly childhood malignancy(56).

A Data sharing agreement specification for compliance and interoperability of access to data:

Despite nearly four decades of single-centre and non-randomized trials, there has been minimal improvement in the survival rates of patients with Diffuse Intrinsic Pontine Glioma (DIPG) (51). Consequently, there is an urgent need for international collaborations to share data and explore new treatments, prognostic factors, and other significant advancements. To address this need, the European Society for Paediatric Oncology (SIOPE) Brain Tumour Group established a DIPG registry and image repository to collect data on DIPG patients (15). However, stringent privacy regulations and high non-compliance fees pose obstacles to seamless data sharing among stakeholders. The automation and enforcement of privacy policies could potentially mitigate non-compliance resulting from intentional or unintentional violations of these rules. Unfortunately, current data-sharing infrastructures and policy reasoning mechanisms somewhat fall short of providing the necessary languages and methods to enforce privacy rules, particularly concerning legal norms such as Data Sharing Agreements (DSAs) and privacy regulations (e.g., the GDPR).

In this project, we utilize a domain-specific language called eFLINT to articulate data-sharing policies expressed in DSAs (109). The eFLINT language, developed to formalize legal norms, proves suitable for specifying data-sharing rules, conditions, and obligations outlined in DSAs and privacy regulations. The extensions made to the eFLINT language enable the connection of higher-level and abstract privacy policies to lower-level system policies, such as read and write access rights (108).

A purpose-based access control mechanism:

Existing conventional access control models do not possess the required mechanisms to articulate and enforce data access and usage policies outlined in privacy laws and contractual agreements. In our research, we introduce an access control model designed to streamline the granting of researchers' access to datasets based on their research objectives. At the core of this model lies the definition of abstract purposes at a higher level, represented as system-level actions or sequences of actions, which form the foundation for purpose-based access control policies.

Develop a user interface :

The current user interface for the DIPG use case was developed by the AMdEX field lab project. The prototype can be used by all stakeholders, a researcher and the DIPG executive committee to submit project proposals and approve project proposals respectively. Additionally, our main objective, to allow researcher access to data based on the approval of a project proposal, has been implemented on the prototype.

2.4 Algorithmic real-time response

A DHT involves collecting and analysing patient data in real-time. Predictions of treatment effects for individual patients can be improved in real-time as the amount of available information keeps increasing. For these predictions to be useful, adequate information about their uncertainty and rationale should be provided to patients and clinicians.

Unfortunately, there are currently many challenges with the way statistical learning and hypothesis testing techniques can be applied in healthcare, and merging real-time response with an adequate confidence measure is one of them. When we peek continuously at our results throughout an experiment ("real-time" analysis), biased and misleading results are retrieved with classical hypothesis testing methods (113). Similar problems arise when we want to combine insights collected over multiple healthcare facilities, or when we want to adapt our experiments post-hoc (48; 100).

A novel framework designed to address these challenges effectively is known as Safe Anytime-Valid Inference (SAVI) (94). This framework enables the dynamic analysis of studies, accommodating changes in their designs during data collection and analysis, thereby avoiding the previously mentioned issue of producing misleading results. SAVI tests can also be extended to estimation, providing anytime-valid confidence intervals that ensure robustness at any point during a study (52).

Furthermore, the combination of multiple SAVI tests into a comprehensive test is straightforward, particularly useful when researchers opt to share only summary statistics instead of an entire dataset. SAVI tests demonstrate flexibility in handling heterogeneous datasets collected from various institutions, accommodating dynamic changes such as switching outcome measures or predictors when data sources evolve.

Our ongoing work involves developing models capable of capturing the confidence of intervention outcomes for small patient groups, ensuring guaranteed error bounds on the model (103; 104). The resulting patient-tailored advice can be presented through an application integrated into the clinical user interface. This application allows dynamic adaptation of input and outcome measures by all key stakeholders: the patient, clinician, and researcher. Notably, the SAVI test or confidence results from such an application can be seamlessly combined with SAVI results from other healthcare centres without compromising the reliability of the estimates.

Thus far, we have addressed inferential models for ongoing studies conducted in real-time, but we also acknowledge the significance of historical observational data in constructing adaptable models. As advancements in data infrastructure and data sharing within healthcare continue, a wealth of such data will become increasingly available. Nevertheless, when working with observational data, the identification of treatment effects may be obscured by confounding variables or

biases stemming from potential fallacies in study design, which extend beyond simple statistical significance. This underscores the necessity for learning models to offer a level of explainability, where the rationale behind the model's decisions is transparent to clinicians, healthcare researchers, and patients (31).

To address this requirement, we explore the utilization of more intricate methods that also possess robust explainability characteristics, such as constraint-based Bayesian networks (18). These models can incorporate multiple predictors and outcome variables of interest, presenting the associations between them in the form of a directed acyclic graph. Through this resulting network, groups or even individual patients who are more likely to respond favourably to specific treatment strategies can be identified and highlighted.

In summary, our aim for the algorithmic real-time response within the DHT is to develop relatively straightforward and transparent mathematical models. These models are designed to offer robust recommendations and insights to support decision-making processes and to enhance understanding of clinical diagnostic methods and therapies. The SAVI tests for inference and the network-based models incorporate prior knowledge drawn from expert insights and pre-training of the models, resulting in low computational costs when deploying the algorithms in real-time settings.

Both proposed models offer significant flexibility, allowing for the incorporation of new insights gained during real-time analysis to dynamically update the models. This adaptability permits adjustments to hypotheses tested and predictions made while maintaining the robustness and ensuring estimation error bounds. Additionally, this dynamic nature enables the modification of predictor variables (in cases of missing data) and outcome variables for each case, facilitating the modelling of individual patients rather than relying on a generic "statistical patient model."

2.5 Analysing distributed data at scale

In addition to the imperative for real-time response, another significant challenge confronting DHTs pertains to scalability. Specifically, for more sophisticated machine learning and deep learning methods to achieve accuracy, substantial volumes of data are requisite for training and assessment (120). However, in the healthcare domain, extensive aggregated medical datasets are rarely available due to data being dispersed and stored across numerous independent institutions. Moreover, stringent privacy laws and policies govern data sharing to safeguard both dataset privacy and patient confidentiality.

A viable solution entails decentralizing computation to where the data resides. To accommodate this paradigm shift, two primary adjustments are essential. Firstly, algorithms themselves must undergo modification to facilitate self-training on distributed data. How data is distributed influences the adapta-

tion required by the algorithm. Secondly, the machine learning mechanism itself must embody inherent privacy-preserving capabilities to prevent inadvertent leakage of information through model predictions or other outputs. These two topics constitute the focal points of our discussion in this section.

2.5.1 Distributed learning algorithms

Data can be partitioned either horizontally or vertically among involved parties. In horizontal partitions, each partition encompasses a set of instances, each possessing all the features associated with that instance, exemplified in section 2.3.3. In this scenario, the dataset encompasses patients from diverse hospitals across multiple countries, where each patient's data is solely recorded within one hospital. Conversely, vertical partitions entail instances where features of a single instance are dispersed across various parties, as observed in the CVA use case, where a patient's data is distributed across different healthcare institutions they visited.

The challenges of distributed learning have been addressed by developing methods that contend with this data parallelism, where collaborative models are trained on distributed data while the individual parties retain ownership of their local data. A significant development in this realm is federated learning ([71]), where local parties construct their models that, through iterative model parameter sharing, converge to a single central model. Models developed for horizontal learning exhibit comparable predictive performance to those trained on fully centralized data ([59]). However, distributed learning on vertically partitioned data remains relatively underexplored ([59]), as it occurs less frequently and entails additional complexity: the participating parties have differing feature sets, thus hindering the ability to train the same model and exchange parameters uniformly. Notably, in vertically partitioned data, only one party typically possesses the label that the model seeks to predict. While recent studies have introduced methods to leverage vertically partitioned data for deep learning, these distributed learning solutions encounter challenges related to predictive performance, privacy, and efficiency ([22]; [50])

For this project, we implemented and assessed Vertical Split Learning, a method where a neural network is distributed across locations like the data distribution ([22]), across a range of medical and other applications. Our findings indicated significant variability in predictive performance depending on the specific use case, feature distribution, and models employed ([12]). This underscores the crucial importance of thorough testing of vertically distributed learning techniques and underscores the ongoing necessity for more resilient vertically federated learning methods. Achieving this entails not only adapting distributed learning approaches to maintain predictive accuracy but also integrating effective privacy and security measures.

Given that this project aims to enhance the broad applicability of Digital Health Twins (DHTs), it is imperative to meet the demands posed by various forms of vertically partitioned data, such as those encountered in the stroke rehabilitation use case. Consequently, our objective is to further refine and tailor distributed learning methods capable of leveraging vertically partitioned data while upholding robust predictive performance, as well as privacy and efficiency standards.

2.5.2 Privacy Preserving Machine Learning

In recent years, distributed machine learning methods have gained significant attention for their ability to address core concerns in data analysis, namely privacy and scalability. Moreover, there has been a notable shift in large-scale, big-data computation toward distributed and cloud systems ([57]). The growing maturity of these methods in other domains makes their application in healthcare particularly compelling, given the sector's inherent requirements for privacy and extensive data processing ([93]).

A new research area, Privacy-Preserving Machine Learning (PPML), has emerged to investigate privacy aspects within machine learning. PPML-based models typically employ techniques based on Cryptography and/or Perturbation to safeguard the privacy of data and/or models. Cryptography-based approaches in PPML utilize cryptographic methods and protocols to encrypt data, while perturbation-based methods involve transforming the data, often by introducing noise to the training data, algorithm parameters, or the algorithm output. Additionally, other less explored methods do not fit neatly into these categories, such as Synthetic Data ([121]) and private aggregation of Teacher Ensembles ([89]).

While various PPML methods have demonstrated their effectiveness, each comes with its limitations. To make these methods viable, machine learning models must be adapted to operate on encrypted, perturbed, or otherwise transformed data, while ensuring that the model trained on altered data maintains similar performance and generalizability as a model trained on unprotected data. The additional computational overhead resulting from these measures is a notable downside, particularly in healthcare, where alongside the critical need for preserving privacy, model accuracy and computational efficiency are crucial factors for adoption in the field.

Our objective is twofold: to offer privacy-preserving machine learning methods tailored for medical applications and to establish a framework for constructing a PPML pipeline customized to the requirements of our stakeholders. This framework is designed to ensure privacy preservation in distributed or federated machine learning scenarios. The distributed nature of the data presents new challenges, including bias, class imbalance, potential unreliability of learning parties, and communication costs, all of which necessitate specific attention for our target use cases. This approach facilitates the execution of algorithms detailed in

the next chapter in a distributed, privacy-preserving manner. Simultaneously, it is intricately integrated into a decentralized distributed scheme encompassing policy specification, infrastructure, and orchestration layers to ensure adherence to governing rules and regulations, as well as accurate setup and deployment at stakeholders' sites.

The adoption of PPML within a DHT aligns with the remaining recommendations outlined in (30) regarding mathematical modelling. PPML facilitates the simultaneous processing of disparate, distributed private datasets without compromising privacy. It enables the establishment of private online repositories, including those containing sensitive individual patient data, while ensuring that data processing remains privacy-preserving regardless of the analytical models developed on top. A framework providing a unified PPML foundation facilitates the seamless integration of phenomenological and mechanistic models by shifting the focus from data leak prevention to the development of hybrid integration methods and strategies. Furthermore, it mitigates privacy incidents when identifying relations across scales, potentially leading to the development of homogenization and distribution strategies for the construction of a theoretical framework for scale separation analysis.

2.6 Conclusion

In the EPI project, we shed light on some of the data-sharing challenges. We focus on solving three use cases involving consortium members. The use cases serve as exemplary applications of personalized medicine and help us to build applicable solutions. Subsequently, we introduced health data processing solutions (real-time response statistical learning) to provide robust personalized recommendations. Data in the real world is distributed, so we investigated distributed learning and various methods to include privacy-preserving techniques.

In the next chapter, we will introduce automated policy interpretation, management, and enforcement to enable data usage abiding by set policies and defined purposes. On a lower level, we will focus on addressing infrastructural heterogeneity and security-based challenges by formalizing the data-sharing framework built with BRANE, and the security service orchestrator. Additionally, we will implement BRANE PoC; which is a distributed research platform; to build and run said workflows irrespective of the heterogeneous nature of the underlying computing resources. This is further detailed in the next chapter, where we discuss all the EPI Framework's components; including BRANE. We will discuss related work and data-sharing framework efforts in the later chapters.

Altogether, the EPI project efforts bring us closer to the creation of Health Digital Twins, a necessary component in enabling personalized interventions. Still, the EPI experiences and insights transcend the medical domain. A major ambition of the EPI project, and in general of the Commit2Data funding scheme

to which the project belongs, was to create long-term and broadly applicable results.

Chapter 3

An overview of the EPI Framework

Robust security frameworks, encrypted communication channels, and advanced access controls are essential components in building a secure data-sharing ecosystem that fosters collaboration while safeguarding patient interests. In this chapter, we develop a framework to combine data analytics, and health decision support algorithms to create personalised insights for prevention, management, and intervention to providers and patients. We design the framework to dynamically provision workflow requests of different healthcare use cases according to set data policy, and network and security requirements. The framework proposed provides the means to build and run distributed frameworks across healthcare institutions within the consortium while reasoning about policies and enforcing network rules. We showcase the EPI Framework's components and discuss the PoC built to deploy the EPI Framework functionality. In this chapter, we partially answer the question:

RQ1: "How can we address open challenges in the context of policy, security, and computing data sharing via building a dynamic infrastructure framework to deploy DHT use cases"

This chapter is based on:

- **Jamila Alsayed Kassem**, Corinne Allaart, Saba Amiri, Milen Kebede, Tim Müller, Rosanne Turner, Adam Belloum, L. Thomas van Binsbergen, Peter Grunwald, Aart van Halteren, Paola Grosso, Cees de Laat, and Sander Klous. Building a Digital Health Twin for Personalized Intervention: The EPI Project. In Commit2Data. Open Access Series in Informatics (OASIcs), Volume 124, pp. 2:1-2:18, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2024) https://doi.org/10.4230/OASIcs.Commit2Data.2

3.1 Introduction

With the exponential growth of health-related data, there is a pressing need for secure data-sharing mechanisms to unlock its full potential in advancing personalized medicine and the development of Digital Health Twins (DHT). The accumulation of diverse patient data, including electronic health records, genomics, medical imaging, wearable devices, and lifestyle information, has paved the way for data-centric health applications. However, harnessing the power of this vast and varied data requires secure and efficient mechanisms for data sharing among healthcare providers, researchers, and other stakeholders. Additionally, secure and efficient data sharing allows researchers and developers to train advanced machine learning algorithms and artificial intelligence models, enabling predictive analytics, precision medicine, and the creation of DHT.

In the rapidly evolving landscape of data-driven innovations and collaborative research, the need for robust frameworks that facilitate efficient data sharing and interoperability has become crucial. This chapter introduces the design of the EPI Framework – an innovative and comprehensive solution aimed at addressing the challenges associated with seamless data exchange across diverse domains. The EPI framework addresses several challenges to digital twin applications in the healthcare domain, such as: 1) strict health data sharing policies often lead to data being locked in silos, 2) legal, policy and privacy requirements make data processing increasingly more complex, and 3) significant limitations on infrastructure resources may apply. The framework comprises three pivotal components, each meticulously crafted to streamline the data-sharing process.

The first cornerstone of the EPI Framework is the Policy Reasoning Tool, an automated approach to formalize, interpret, and manage data-sharing policies. The EPI framework leverages eFlint's capabilities, and the Policy Reasoning Tool empowers users to navigate the complex landscape of data sharing with an emphasis on adherence to policies, ensuring ethical, legal, and regulatory compliance. This tool serves as the governance backbone, fostering a secure and accountable environment for data exchange within the framework. We expand more on this in Section 3.2.

Complementing the Policy Reasoning Tool, the second component is BRANE – a Pipeline Computing Orchestrator. Engineered to define and manage distributed workflow pipelines, BRANE optimizes computational resources, enhances data processing efficiency, and does so in compliance with set policies. Section 3.3, introduces BRANE's key features and functionalities, in addition to implementation efforts. The third and final pillar is the Bridging Function Chain (BFC) Orchestrator – a virtual security and network functions orchestrator designed to bridge the gap between security areas across data-sharing endpoints. Alongside BRANE, the BFC orchestrator is utilized to enforce security policies and ensure endpoint reachability for a successful data-sharing use case deployment. We introduce the BFC orchestrator in more detail in Section 3.4. In this chapter, we

lay the groundwork for a robust, flexible, and scalable data-sharing framework that caters to the evolving needs of modern collaborative research, governance, and decision-making.

3.2 Automated policy interpretation, management and enforcement

Current approaches to accessing healthcare data for research purposes can be complicated due to strict privacy regulations and contractual agreements such as data sharing agreements. Privacy regulations and data sharing agreements are usually written in natural language, which can lead to ambiguous and inconsistent interpretations (11). Resolving these challenges demands suitable policy specification languages to specify policies. The main objective of this work is to automate data access requirements from privacy regulations and data sharing agreements (DSAs) to reduce the complexity of data access for research purposes and enable the enforcement of different sources of legal norms via access control mechanisms.

3.2.1 Data sharing agreements and access control

The General Data Protection Regulation (GDPR) states the principles and rights that must be met for processing any personal data (26). These principles are lawfulness, fairness and transparency; purpose limitation; data minimization; accuracy; storage limitation; security and accountability. The GDPR provides strict privacy norms and there is no (access control) mechanism that enables the enforcement of all such norms. These challenges become more prevalent when several sources of legal norms (e.g., GDPR and DSAs) are simultaneously applicable, as is often the case with medical research. Additionally, when data is merged from several sources and different controllers are allowed to author their own policies that dictate for each source, it can cause policy conflicts. The gap between legal requirements and their technical realization is due to the lack of machine-readable representation of privacy policies.

The goal is to formalize privacy policies and to enable their enforcement via (novel or existing) access control mechanisms. In the initial stage of this work, different policy specification languages were analysed to determine which of them met the privacy policy requirements from GDPR and DSA of the DIPG Registry use case. The Open Digital Rights Language (ODRL) (54) and the eXtensible Access Control Markup Language (XACML) (99) were concluded not to be sufficiently expressive. The analysis of ODRL is reported in (70). Instead, we are applying the eFLINT language, originally developed within the SSPDDP

project[1], to formalize GDPR and DSA articles relevant to the Diffuse Intrinsic Pontine Glioma (DIPG) use case.

The eFLINT language is a domain-specific language that was specifically designed to formalize legal norms (109). The eFLINT language extensions enabled us to interconnect higher-level privacy policies such as those expressed by the GDPR and DSAs with system-level policies such as those specified by access control policies (108). Additionally, we formalized the access request workflow of the DIPG registry use-case, a researcher submitting project proposals and approving proposals and the duties that follow as a result of these transitions. Finally, we demonstrate how actions described by the GDPR such as "collecting personal data" can be synchronized with actions from DSA such as "making data available" to the registry. This approach allows for the modular specification of norms by permitting re-use between specifications as well as defining alternative specifications.

3.2.2 Purpose Based Access control mechanism

Within the EPI Framework, we propose an access control model that simplifies the process of granting researchers access to datasets based on research purposes. Article 5(1(b)) states that personal data shall be collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes. Additionally, If personal data is collected for more than one purpose and these purposes are related, then the GDPR allows for processing to take place under an "overall purpose" under which a number of separate processing operations can take place. Therefore, the purpose limitation principle minimizes the risk that might arise when personal data is processed by confining the possibilities of its usage by limiting instances of lawful processing.

Our purpose-based model is based on these requirements. The main component of this model is the specification of higher level abstract purposes as system level action or sequences of actions which are the basis for defining purpose-based access control policy. We model action relationships based on the purpose specification requirements from the GDPR and express constraints over these relationships using the eFLINT language. The goal of this work is to design and implement an access control model that allows access to data based on the purpose of research plans.

[1]The SSPDDP project (628.009.014) is part of the NWO program Big Data: Real-Time ICT for Logistics

3.3 BRANE - A heterogeneous, distributed research platform

Different DHT functionality requires different physical infrastructures to be deployed. On a software level, they have different hardware- and software requirements, such as dependencies on accelerators or specific software packages; and on an application level, they need different data flows and might impose different security requirements. Therefore, there is a need to dynamically map a logical infrastructure dictated by a DHT functionality onto a physical pool of resources. This pool of physical resources is often heterogeneous and dispersed as well. The exact composition varies but may consist of resources in the cloud, high-performant grid computers or even small, embedded system devices that operate only locally. This variety introduces interoperability issues on the software level (different software stacks, transfer protocols or granted system privileges), hardware level (processor architectures, availability of accelerators, networking capabilities) and administration level (different costs, usage quotes or service-level agreements). Furthermore, heterogeneous trust relationships between resource providers may exist (86). This restricts how data can flow between resources, or sometimes even if the provider of a resource is willing to do a task at all if this violates their own trust- or security policies. This issue becomes even harder when some of the policies involved are sensitive themselves and must thus be kept private; it may be, for example, that a dataset can only be processed by a resource if a patient has given consent to do so, where this consent itself already reveals sensitive information about the patient.

Traditionally, the mapping of a logical infrastructure on a physical pool of resources is done (semi-)manually. However, this approach does not scale in the case of DHTs, as both ends of the mapping are dynamic: new DHT functionality may be introduced or adapted, resources may be introduced or removed and the policies of existing resources may change. Hence, a mechanism is required to perform this mapping automatically and dynamically. To solve the software and hardware interoperability issues, the BRANE Framework (106) is introduced as the mechanism to provide the logical infrastructure to a physical resource pool mapping. It frames DHT functionality as *workflows*, which are compositions of elementary functions to form (data) pipelines that implement the desired behaviour. These elementary functions, in turn, are implemented by *packages*, which are containerized pieces of software such as data preprocessing steps or (parts of) algorithms that are executed on the resource pool. The runtime of the framework can then orchestrate these containers based on the requirements and dependencies laid out by the workflow and provide the mapping; this act is called *planning* a workflow. A visualization of this process is given in Figure 3.1.

To handle the administrative-, privacy- and trust-related interoperability issues, the BRANE framework is extended with a notion of *policy* (36). These

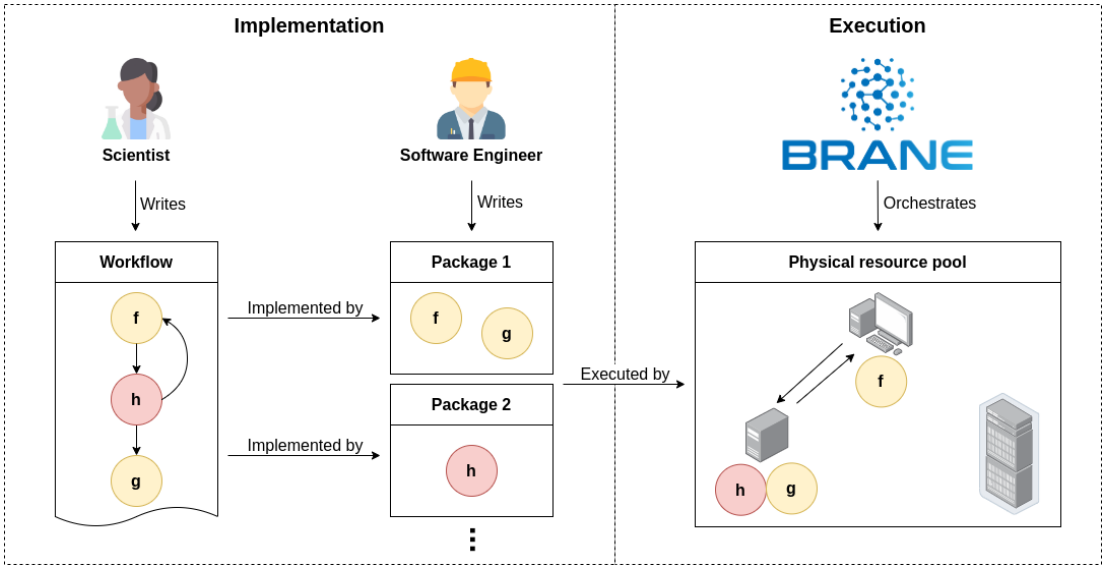


Figure 3.1: **Conceptual role of BRANE.** During the implementation of a use case, a scientist writes a workflow that composes elementary functions (`f`, `g` and `h` in this example). These elementary functions are implemented by packages, which are written beforehand and made publicly available by software engineers. At runtime, the BRANE runtime understands the dependencies between the functions and can map them onto a physical pool of resources.

policies are expressed on a per-resource level, which effectively makes policies representative of a resource owner's wishes within the system. By consulting with the policy both during the planning of a workflow and during execution, the policies restrict what a resource is used for and what happens to the resource's data, as well as express administrative limits or costs on the resource. A key feature of this notion of policy is that the policies are abstracted away behind a service, the *checker service* (Figure 3.2). Using an interface like this allows resource owners to make their policies arbitrarily complex since the system has no requirements on how they are implemented. For example, they can use languages like eFLINT (109) that can capture norms, allowing policies to express legal concepts like GDPR. Moreover, addressing the need for private policies, the interface allows every participant to hide as much of their policy as desired. This can range from "metadata" of a policy, such as the name of the person who gave consent, to the policy rules themselves; although the latter requires advanced reasoning about what policy information is leaked through the interface. This ability to keep policies private is what sets BRANE apart from similar workflow systems, like (110) or (29).

In summary, we present an automatic mapping of the logical infrastructure dictated by DHT functionality to a physical pool of resources using the BRANE infrastructure. It frames the functionality as workflows and packages and com-

plements that with high-level constraints on that mapping by introducing (potentially private) policies that are hosted on the resources themselves. To enforce lower-level policies related to (network) security, a more complex extension to BRANE is required and described in the following section.

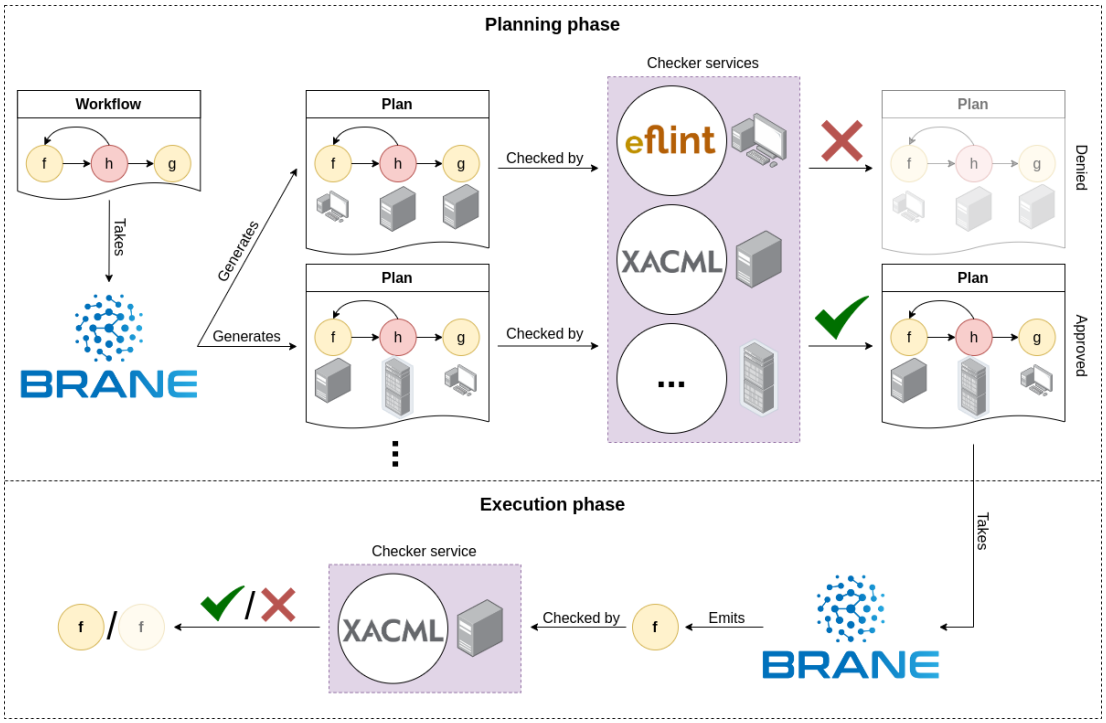


Figure 3.2: **Policy enforcement in BRANE.** There are two phases when executing a workflow: in the first, the *planning phase*, the runtime maps tasks in a workflow to resources that will execute the tasks. Here, checker services (and thus policies) act as filters to separate plans they would allow from plans they would deny, preventing the system from having to execute every possible plan. Then, during the *execution phase*, the runtime starts traversing the planned workflow and attempts to execute each task on the resource it was planned on. The checker of that specific resource examines the task again, to prevent any staleness introduced by the arbitrarily long delay between the planning check and the execution of that specific task.

3.4 Distributed Analysis Infrastructure

Currently, "working in silos" is still dominating the healthcare industry, but data-sharing between health domains is key on the road towards DHT. We aim to lay the foundation of distributed infrastructure and utilize it in deploying the previously mentioned use cases. In the context of this project, we define a data-sharing

framework that allows parties in specific (research) networks to share patient information securely in an ad-hoc way in the course of treatment or a specific research question, as well as structurally for long-term studies while maintaining control and tractability/suitability to data controllers and subjects at the source.

Developing a generic solution to comply with the dynamic security policies is difficult, mainly because the infrastructure between different healthcare domains is heterogeneous in terms of computing, networking and storage functionalities. Moreover, when various use cases have to be deployed (e.g. for running an ML algorithm, monitoring, sharing EHRs, etc.), different requirements or rule sets (policies) apply (e.g. privacy related) and have to be enforced to ensure proper control over the data flow/ usage. The infrastructure and security models need to support these different applications and therefore need to be adaptive to be able to support dynamic and often application-specific sets of requirements.

The main design properties we consider while setting up a secure, collaborative networked environment ([63]) between different EPI parties are:

1) the low-level security and network policies formalizing and enforcement,

2) compatibility with higher-level policies, and dynamic workflow schemas,

3) evaluating data-sharing domains, and adaptively provision network service chain to maintain security and quality of service,

4) reacting to any policy change while flexibly trading and routing packets via the chain via proxies (different implementation presented in ([67])).

The proposed framework considers the intended data-sharing use case, and the policies associated. We map the mandated network services to the defined enforcement primitives: *filter* traffic and/or *transform* traffic. The framework dynamically provisions these services by placing the functions on available N-PoPs (Network Points of Placements), assigning the service requests to the running function, and routing traffic along the function's chain to enforce a policy. We *optimize* provisioning decisions to maximize the quality of service based on the infrastructure state, the use case's requirements, and the CPU profiles of services ([61]). The hyperparameters are tuned to prioritize resource utilization or latency in an effort to comply with the performance requirements. Three provisioning tools are used: a greedy heuristic approach, Deep Q-Learning (DQL), and a Heuristic-boosted DQL (HDQL).

We manage and orchestrate virtualized networked services on top of the existing client's infrastructure to increase the security level of the communication channels and enforce policies. A cloud-native network orchestrator is defined on top of a multi-node cluster mesh infrastructure for flexible and dynamic containerized function scheduling. The expected challenges include verification of rules with laws and international policies, integration with legacy systems, and

acceptability of the framework by health institutions. The latter includes providing the health institutions with sufficient oversight and control over the sharing and usage of their data so that they are confident that this exchange is compliant and controllable. We address these challenges by working closely together with the responsible entities (IT departments, ethical boards, and data privacy officers) in the associated hospitals. The approach we will be taking is to formalize a logic to automate infrastructure setup per application scenario by utilizing virtualized services hosted by a bridging node. The framework runs a middleware in a virtualized manner and offers network services to secure data sharing and consider policies.

3.5 The EPI Proof of Concept

In order to experimentally validate the work presented in this paper, a proof-of-concept (PoC) has been deployed in cooperation with various organizations in the EPI consortium. The goal of the PoC is to create an environment in which federated applications may be executed which involve one or more 'local' processing steps followed by a centralized 'global' step that aggregates the local results. In the PoC, local steps are executed by St. Antonius and UMC Utrecht, each having a part of a horizontally split dataset. To test the EPI use cases, as well as our methods of analysing distributed data at scale, we use a dataset that has been previously collected in both hospitals with characteristics and treatment outcomes of patients who received electroconvulsive therapy. Results of the local computations on this dataset are sent to SURF, which plays the role of a trusted third party that can aggregate the results (see figure 3.3). SURF also hosts the BRANE runtime that acts as an orchestrator.

The PoC is used to assert that the framework discussed in sections 3.3 and 3.4 can support the resource pool hosted by the three participating organizations. Specifically, the resources provided are heterogeneous in three of the aforementioned dimensions: they differ in hardware capabilities, software stacks and trust relations. Regarding trust: only the hospitals are allowed to see only their own ECT dataset. SURF is only allowed to see the local results produced on each site and the global results they produce. These rules are ideal for experimenting with automated policy enforcement (section 3.2). The setup also validates whether the framework can operate within the administrative and security restrictions imposed by the security requirements of the hospitals. Most notably, virtual private networks (VPNs) ([112]) are used to safeguard the data as it travels over the public network, network access is restricted to the outside world and some domains only offer restricted rights to the framework runtime.

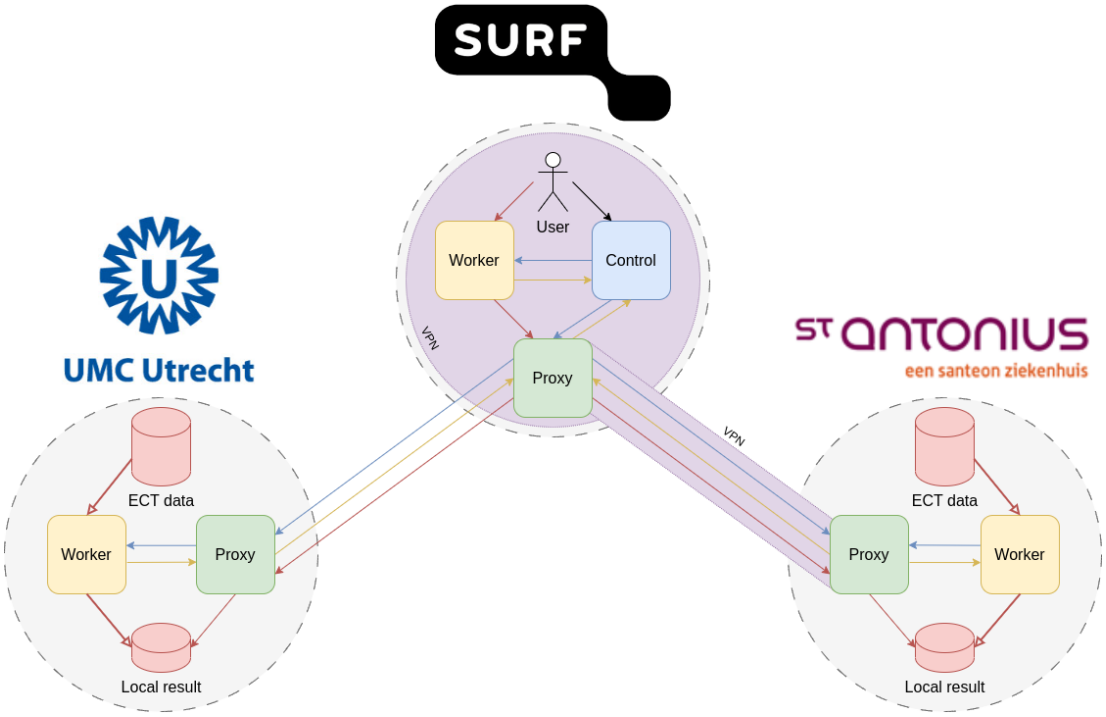


Figure 3.3: **Schematic overview of the PoC setup.** Three domains participate in the proof-of-concept, each of which has its own resources hosting their part of the framework: each hospital (UMC Utrecht and St. Antonius) has local instances of the ECT dataset, and hosts a worker node to perform local computations; SURF, meanwhile, acts as an entry point, and hosts a worker to aggregate the local results into a global one. Domains are contained in each dotted sphere, where each sphere shows the relevant components for that domain. The arrows indicate network traffic between the domains, where the direction of the arrow indicates the direction of the initial message.

3.6 Conclusion

In this chapter, we highlighted the EPI Framework functionalities, components, and how these interact. We defined the EPI Framework that runs the EPI healthcare use cases, adhering to a set of policies. To do that, we delegate different functions into three components: eFLINT policy reasoner, BRANE workflow orchestrator, and Bridging Function Chain (BFC) orchestrator which handles reprogramming the infrastructure/request. The EPI components are put in a collaborative architecture to address data-sharing challenges on many levels: application level where we need to run different workflows (potentially distributively) regardless of the heterogenous computing capabilities of the node, policy level where we need to manage and adhere to data-sharing agreements and laws, and reprogramming the overlay network of services to enforce some obligations and rules.

The set of policy and security rules are use case dependent. Hence, in the next chapter, we propose the area logic model that aims to automate infrastructure orchestration.

The alignment between allowed, and feasible data flows at the infrastructural level may not always be straightforward. Theoretically, policies dictate the permissible actions on data, while at the practical infrastructural level (specifically within the BRANE and BFC orchestrator domains), the workflow of data pipelines actually is deployed. In the next chapter, we delve deeper into this dynamic by introducing the Area Logic Model. This model serves as an automated framework for reasoning about the intersection between allowed and feasible data flows, bridging the gap between policy-driven ideals and the operational pipelines. We will present related work, and draw comparisons between similar data-sharing framework efforts in the next chapters.

Chapter 4

EPI Policy Resolution: Area Logic Model

The EPI services run over adaptive infrastructures, which provide more flexibility to accommodate different data-sharing requests. This chapter introduces the logic model behind it, intending to support novel health services over programmable infrastructures. A single static infrastructure cannot support different healthcare applications and collaborate data between stakeholders. This infeasibility is associated with the heterogeneity of infrastructures/ healthcare domains, namely the fact that computing/ networking/ storage capabilities differ from one domain to the other. We propose a dynamic programmable infrastructure that can bridge functionality gaps, hence providing data availability and interoperability when sharing data between heterogeneous infrastructures.

We define the formalism of the logic model to deduce feasible data movements between EPI endpoints and possibly satisfy a data collaboration request. We reinforce the infrastructure orchestrator's logic model by introducing the algorithm running on this federated system and further provide three healthcare use cases' walk-throughs. We evaluate our model according to three relevant parameters, performance, feasibility, and aggregation power, and we can conclude that our model supports the required interoperability between the EPI partners. The framework assigns nodes into a hierarchy of areas, initializes channels, considers collaboration rules and aggregates bridging functions to control the information flow in a sub-infrastructure. In this chapter, we partially answer the question:

RQ1: "How can we address open challenges in the context of policy, security, and computing data sharing via building a dynamic infrastructure framework to deploy DHT use cases?"

This chapter is based on:

- **J. A. Kassem**, C. de Laat, A. Taal and P. Grosso, "The EPI Framework: A Dynamic Data Sharing Framework for Healthcare Use Cases," in IEEE Access, vol. 8, pp. 179909-179920, 2020, doi: 10.1109/ACCESS.2020.3028051.

4.1 Introduction

In the landscape of health data-sharing frameworks, the convergence of policy-driven ideals and the operational realities of data pipelines within dynamic infrastructures often presents a challenge. While policies theoretically define the allowed data flow, the practical execution within infrastructural domains, especially within BRANE and BFC orchestrators, can introduce complexities that impact the alignment of allowed and feasible data flows. This chapter takes a closer look at this by introducing the Area Logic Model.

This chapter contributes to the existing literature by introducing a novel methodology designed to determine achievable data-sharing workflow archetypes within dynamic infrastructure landscapes. Our approach defines the EPI areas, an abstract grouping of EPI endpoints according to security and infrastructural attributes. Around that, In Section 4.3 we build an area logic model that aligns with and maps to possible data flow within complex and dynamic infrastructural settings.

To operationalize our methodology, in Section 4.4 we implement a robust algorithm dedicated to the creation of EPI areas. The evaluation of our algorithm is a crucial aspect of this chapter, as we assess in Section 4.5 the performance against relevant parameters. In Section 4.6 we analyse the effectiveness of our proposed methodology and the practical implications of achieving data sharing within heterogeneous health data ecosystems.

Furthermore, in Section 4.7 the application of our methodology is demonstrated through the utilization of a logic model, specifically tailored for EPI areas, in addressing EPI use cases. By applying the logic model to real-world scenarios, we seek to illustrate the adaptability and efficacy of our methodology in diverse health data-sharing contexts. As a result, in this chapter, on the road towards secure medical data sharing, we automate the aggregation of high-level policy with low-level infrastructure capabilities for seamless and compliant workflows.

4.2 Related Work

The latest advancements in the field of informatics inspired much of the research utilising said technologies in the healthcare domain. Such development is cur-

rently essential due to the impractical traditional data-sharing methods. According to surveys (90), roughly 15% of patients visiting a doctor were asked to supply their radiology report personally (like on a DVD), and another 5% had to redo their tests leading to more radiation.

Healthcare providers realise that and are migrating towards the usage of electronic medical records (EHR). There is a broad consensus that sharing data in the medical spectrum can be beneficial in terms of cost-efficiency, preventing redundancies, cooperation between stakeholders, and reliable research by accelerating innovations and discoveries. Achieving secure health data sharing can result in an efficient and effective healthcare cycle managed by the patients/healthcare stakeholders.

Health data-sharing frameworks exist, but they are catered to satisfy a single, specific use case, which makes the architecture rigid and less suited to support different applications. These frameworks do not address the different capabilities present in all network endpoints and assume identical operational possibilities on all participating nodes. This is far from the current state-of-the-art in computing infrastructures.

In this section, we aim to highlight some of the literature. The way that we gathered the literature is as follows. For each query on Google Scholar, we selected the top result. We queried "secure sharing infrastructure", and the top result was (96). The search results were further filtered with a 2018+ time frame to represent more recent research (91). We queried "secure data sharing infrastructure", and the top result was (95), then added a time filter of 2018 which gave the top result (91). We queried again "health data secure monitoring" with top result (101), and got (47) as a top result for the results published after 2018. The mentioned papers referenced interesting literature like (117) (39) (14) (81).

Some frameworks provide interoperability of data but for a single use case. Roelofs *et al.* (95) discuss an open-source infrastructure to share medical data internationally for radiotherapy studies. They provide interoperability of data by running a multicentric data mining. While Sartipi *et al.* (96) introduce an infrastructure that integrates PACS (Picture archiving and communication system) and HL7 (Health level 7) and aims to have a homogeneous, and internationally accepted EHR (Electronic health records).

Other frameworks rely on static and specific technologies or devices. In more recent papers, frameworks leverage blockchain as a single infrastructural data-sharing solution. Patel (91) favours an established consensus on a blockchain ledger over third-party intermediates. Another framework utilising the same tools is MeDShare (117). The system integrates smart contracts and access control protocols to provide a tamper-proof ledger of health data. MedBlock (39) compares with the MeDShare system, and shares data via blockchain. Unlike MeDShare, MedBlock does not maintain a ledger of audited behaviour, instead, it provides an information management system. Moreover, Thilakanathan *et al.* (101) proposes a platform that is based on cloud data services. The platform adds a layer of

security by proposing a security protocol with ElGamal and proxy re-encryption key exchange schemes.

A different application of medical data sharing is secure monitoring. Anoop *et al.* (14) propose streaming an enormous chunk of medical data collected through wearables using BSN (Body Sensor Network) to provide real-time monitoring of the patient's health status. Manogaran *et al.* (81) design a new architecture of IoT environment to store, process, and share big data. Griggs *et al.* (47) build a blockchain-based system to remotely monitor the patient's status. The system uses smart contracts on a private Ethereum network to handle medical information.

Roelofs et al. (95) propose an open-source infrastructure to share medical data internationally for radiotherapy studies. The endpoints of the system are the international institutes to run multicentric data mining for radiotherapy research. They built a research data model, queried needed data, and then ran a pseudonymisation code. SNOMED is used as a medical dictionary and is further synchronized by the DICOM (Digital Imaging and Communications in Medicine) mechanism in a central model. DICOM is the international standard to transmit, store, retrieve, print, process, and display medical imaging information. The security is addressed by setting up a private VPN to share data. Sartipi et al. (96) introduce an infrastructure that integrates PACS and HL7 to provide an internationally accepted EHR. They use the XDS-i protocol to cooperate multi healthcare agents and OpenID-0Auth for authorisation and authentication. In their proposal, the cooperation of agents supplies action-based mechanisms for access control. Moreover, security is also addressed using a policy that detects behaviour patterns.

Patel (91) leverage an established consensus on a blockchain ledger over third-party intermediates. This development provisioned a radiology research ledger with cross-domain image sharing. The system security and privacy rely on the asymmetric key-pair mechanism and patient access permission. MeDShare (117) is another framework utilising blockchain tools by providing a trustless sharing of medical data between cloud service providers. The system integrates smart contracts and access control protocols to provide a tamper-proof ledger of data behaviour. MeDShare infrastructure monitors entities and provides an auditing ledger. Based on what has been done, offending or suspicious entities are revoked access to medical data on cloud repositories. MeDShare assumes that medical data is stored on a cloud and runs on top of that to supply data provenance, monitoring and auditing, and access rights and control.

MedBlock (39) compares with the MeDShare system, and shares data via blockchain. Unlike MeDShare, MedBlock does not maintain a ledger of audited behaviour, instead, it provides an information management system. The architecture of the systems includes the user uploading his medical data to a processing layer of hospital servers. After that, the encrypted medical records are added to the ledger, from which the user can query/retrieve data. Securely communicat-

ing with the help of a certificate authority that will distribute keys. The security and privacy of the system are delegated to the blockchain security and consensus mechanism, CA, and the access control mechanism.

More infrastructure papers covered a different application of medical data sharing and secure monitoring, e.g. (14). The application is concerned with streaming an enormous chunk of medical data collected through wearable using BSN in the effort of real-time monitoring of the patient's health status. (101) proposes a platform that is based on cloud data services, but adds a layer of security by proposing a security protocol with ElGamal and proxy re-encryption key exchange schemes. The paper seems to address the problem of retracting authorisation of the health stakeholder by the user at any time. That is done by simply retracting the "data consumer" key entry from the user key database. Due to that, the user's right to receive or access data is revoked optimally without any delays and puts on the user too much responsibility by re-assigning new keys to all users and having the users up to date with the valid keys. It seems that the security protocol depends on Cloud and encryption algorithms, which by itself can cause some security issues. Other than that, the framework seems to assume that the DSS (Data-Sharing Service) is trusted, which begs the question, will that be enough for hospitals and other health stakeholders to share their data and collaborate?

(81) designed a new architecture of IoT environment to store, process, and share big data. "Meta fog redirection" and "grouping and choosing" are mentioned to be two sub-architectures of the design. The first is using Apache Pig and Apache HBase to store collected sensor data. The latter is then used to integrate fog and cloud computing. The integration of fog computing has security implications for the system. It utilises the bandwidth, response time, and scalability. It relies on the PKI key management scheme to secure data sharing and control access.

(47) build a blockchain-based system to securely and remotely monitor the patient's status. The system uses smart contracts on a private Ethereum network to handle medical information in a fast and secure manner. The communication starts with the medical sensor streaming into a smart device that calls the function of a smart contract that processes the data. If the data retrieved is worrying, an alert is sent to the user and the hospital. If not, the system continues to audit and commit all activities to the ledger. The system delegates security to traditional encryption and IP protocols, and the consensus of signatures.

Table 4.1 shows a comparison between these frameworks according to a set of considered features. In light of the frameworks discussed, some elements, or combinations of elements, might be lacking. For instance, some proposals consider security on the one hand but don't address relevant laws and auditing. On the other hand, other frameworks address interoperability and data model usage across different sectors, hence pre-process heterogeneous resources, but don't offer access control and security to protect the sensitive data. More importantly,

all these frameworks are catered to satisfy a specific use case, which makes the architecture rigid and hard to support different use cases.

Framework	Tools	Use Case	Security considered	Access control	Data audit	Laws	Dynamic	Interoperability
[96]	PACS, XDS, OpenID	Medical images sharing	✓	✓	✓	×	×	✓
[95]	SQL, DICOM, Key scheme	Radiotherapy research data	×	×	×	✓	×	✓
[91]	Blockchain	Medical images consensus	✓	✓	×	×	×	×
[117]	Blockchain, Cloud	Medical data sharing	✓	✓	✓	×	×	×
[101]	Cloud	Monitoring and sharing data	✓	✓	×	✓	×	×
[39]	Blockchain	Medical data sharing	✓	✓	✓	×	×	×
[81]	Cloud, Fog computing	Smart healthcare monitoring	✓	✓	×	×	×	×
[14]	WBANs	Secure monitoring	✓	✓	×	×	×	×
[47]	Blockchain smart contracts	Remote patient monitoring	✓	✓	✓	×	×	×

Table 4.1: Comparison between frameworks

The majority of the proposed work is application-specific, and all shared resources are exclusively data. These infrastructures are mainly static and offer a "one fits all" standard. None of these frameworks address the data movement requirements for generic use cases. In EPI, we formalise a methodology to support data-sharing requests across providers with heterogeneous resources, which relies on the programmability of the infrastructure.

4.3 The EPI Framework: The Infrastructure Orchestrator

The EPI infrastructure is the collection of all networking, computing, and storage *nodes* provided by EPI parties. The EPI infrastructure orchestrator will cater to any application scenario by building a sub-infrastructure.

Figure 4.1 illustrates a high-level view of the proposed infrastructure orchestrator. An *application scenario* specifies the resources needed, and the data collaboration goals, and is initiated by the collaborating parties (1).

The Information Sharing Agreement (ISA) describes the data sharing policy between parties, *i.e.* a policy is the group of rules governing data movement/usage ([98]). The EPI infrastructure orchestrator queries ISA from the policy management system (2). The received ISA is further translated to a set of rules (3) that are recorded into a log.

The orchestrator uses a logic model (Section 4.3.1) to group nodes into areas that describe feasible data movements between them (4). On top of feasible data movements, the framework also considers the information flow rules log to determine possible archetype mapping (5). The model describing the pattern of data movement is named *collaboration archetype* ([119]).

An *application request* is the actual-requested data movement between parties. After building the sub-infrastructure, application requests are introduced (6). A single application scenario can refer to multiple application requests, and the aggregation of all possible data movements are represented by archetypes. When

an application request maps to an existing collaboration archetype (7), it can be further applied in the sub-infrastructure (8).

An auditing log of the data movement behaviour is maintained (9). The audit log is compared to the rules log, hence providing accountability. (10).

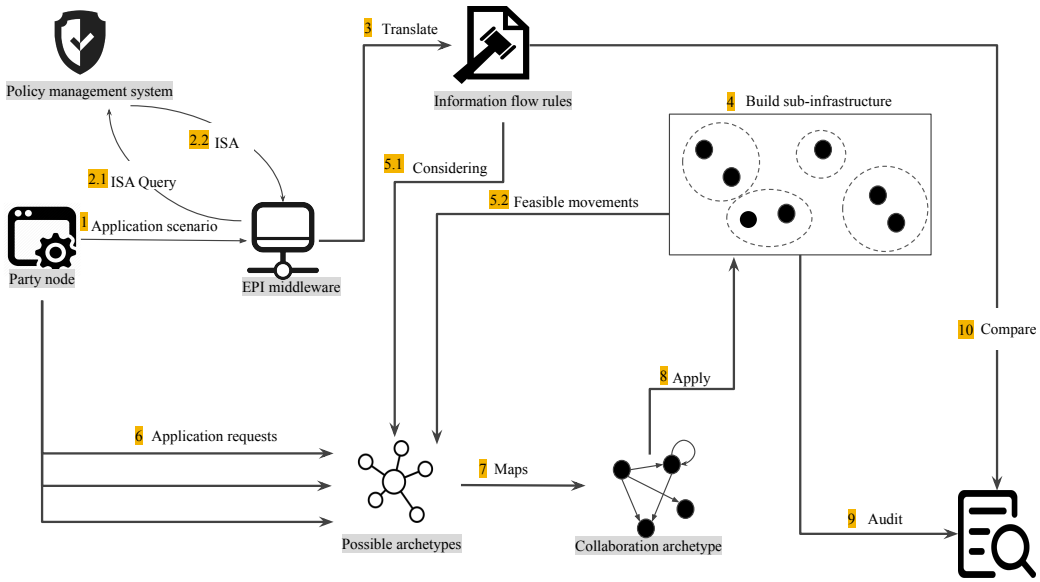


Figure 4.1: A high-level view of the EPI orchestrator architecture after an application setup is initiated.

4.3.1 The Area Logic Model

This section explains the logical model of creating and assigning area nodes, hence creating a distinction of nodes' capabilities. Initially, there exists a clear view of all possible parties (e.g. hospital, research centre, rehabilitation centre) that can partake in or initiate an application scenario setup. We assume that a tool on a higher level of framework fills out an ISA when a set of parties (healthcare domains) initiates an application scenario. The ISA is assumed to consider all requirements under which data collaboration is allowed, and it gives back a set of permission rules. That is further translated into rules, which leaves the issue of actually moving the data. The issue arises due to the data movement, depending on the compatibility and data interoperability between parties.

4.3.2 Sub-infrastructure Initialising

A health domain/provider party assigns a number of nodes which are the endpoints of physical/virtual resources. The first step is to establish "who" can

offer "what". We represent all resource endpoints included in the underlying EPI infrastructure by the set N of nodes:

$$N = \{n_i \mid i = 1, ..., m\}, \tag{4.1}$$

Where n_i represents a single node in the infrastructure and m is the total number of nodes.

Attributes refer to what a node can support in terms of networking, computing, and storage functionalities. Possible attributes can specify software and computing resources: software tools, response scalability, and CPUs. It can also specify security and network resources: identity and access control mechanisms in place, data encryption and key management, data anonymisation, firewalls, and scalable network links. Attributes for storage resources can be maximum data storage, database tools, or structured EHR/ unstructured. Let A be the set of all possible attributes that could apply to a node:

$$A = \{a_k \mid k = 1, ..., d\}, \tag{4.2}$$

Where a_k represents a distinct attribute, and d is the total number of attributes.

Once the application scenario is made clear, then the sub-infrastructure is initialised to support it. A sub-infrastructure is a set of nodes supporting a specific application for a duration of time. Let N_{App} be the set of mentioned nodes, such that:

$$N_{App} \subseteq N \tag{4.3}$$

While as, N is the collection of all the nodes of all parties, N_{App} specifies subset nodes the collaborating parties use in the application scenario. The nodes needed to support the application are determined and the larger set N is filtered down. Each node's capabilities are described by a set of infrastructural attributes, which can be any subset of A. Subsequently, every node n_j in N_{App} is assigned $A_j \subseteq A$.

The model inspects nodes within N_{App} and queries the attributes associated with each node. We define a mapping f_{App} between the set of nodes N_{App} and the power set $\wp(A)$ of A, such that $f_{App}(n_j) = A_j$. Eq. (7.4) notates the many-to-one relation between nodes and attribute sets:

$$f_{App} : N_{App} \to \wp(A). \tag{4.4}$$

f_{App} is used to determine μ, where μ is the set of attribute sets related to nodes in N_{App}:

$$\mu = \{A_j \mid j = 1, ..., ||N_{App}||\} = \{f_{App}(n_j) \mid n_j \in N_{App}\}. \tag{4.5}$$

4.3.3 Creation of Areas

The area abstraction is used to spot heterogeneity between nodes and deduce what movement is supported.

Let Θ_{App} be the set of all areas grouping N_{App} within the sub-infrastructure:

$$\Theta_{App} = \{\theta_p \mid p = 1, ..., l\}, \tag{4.6}$$

Where θ_p is a single area associated with a distinct set of attributes, and Θ_{App} is a partitioning on the set N_{App}.

An equivalence relation on N_{App} is defined such that $n_g \sim n_h$ if n_g and n_h belong to the same θ_p and with $n_h \sim n_g$ denoting $f_{App}(n_g) = f_{App}(n_h)$ or $A_g = A_h$. As in Eq. (4.7), deterministic function α gives the area set Θ_{App} having N_{App} and μ as an input:

$$\alpha(N_{App}, \mu) = \Theta_{App}. \tag{4.7}$$

As a result, areas map to the total resources and endpoints per application setup within a sub-infrastructure and the differences between said nodes. Based on that, supported data movement *channels* can be deduced from this mapping.

4.3.4 Supported Channels Within a Sub-infrastructure

The logic dictates that data movement is supported when Eq. (4.8) is satisfied, where $n_g \Rightarrow n_h$ represents a one data movement support between nodes n_g and n_h. That means that a directional movement from n_g to n_h is supported when the capabilities of n_h (A_h) is the same or a superset of that of n_g (A_g)

$$n_g \Rightarrow n_h, iff A_g \subseteq A_h. \tag{4.8}$$

Supported/unsupported movements are represented by channels that can be utilised via collaboration between nodes. The α output is further processed and translated into a channel matrix by function β

$$\beta \circ \alpha(N_{App}, \mu) = C. \tag{4.9}$$

An adjacency matrix represents all nodes connected by a channel, where a single entry c_{ij} is the channel between the sender node n_i and the receiver node n_j, such that C is a $s \times s$ square matrix with $s = ||N_{App}||$. A single entry c_{ij} is a Boolean value where 0, 1 represent the existence of a channel or the lack thereof, respectively.

4.3.5 Applying Flow Rules

To determine and set up the channels, it is not sufficient by itself to control the information flow within the infrastructure. The application scenario's policy further uses the channels in place to regulate and dictate data movement. The data flow rules R are set in the light of ISA that are placed by a policy and management system. The rules describe all required data movements of an application scenario mindful of the policy. R represents the allowed/denied data movement the application needs to run. Rules are translated into a matrix form to make it easier to aggregate with other variables:

$$\gamma(ISA) = R \tag{4.10}$$

All the rules between the nodes are represented in an adjacency matrix R, such that a single entry r_{ij} refers to the rule between the sender node n_i and the receiver node n_j, such that R is a $s \times s$ square matrix. A single entry r_{ij} is a Boolean value where 0, and 1 represent the denied and allowed data movement rules, respectively.

We differentiate between allowed/denied and supported/unsupported data movements. Ideally, rules restrict further supported movements. As an example, when $n_i \Rightarrow n_j$ is supported, this means that $c_{ij} = 1$. Moreover, if data movement utilising c_{ij} is also allowed, this means $r_{ij} = 1$, hence the rule set aligns with channels.

4.3.6 Bridge Attribute Gaps

In other cases, rules and channels might not necessarily align, which can be an issue in running an application successfully.

The condition $A_g \subseteq A_h$ supports data movement from node n_g to n_h. Otherwise, A_g is not a subset of A_h. In Eq. (4.11), ϵ_{gh} indicates the missing attributes

$$\epsilon_{gh} = A_h \cap \overline{A_g}. \tag{4.11}$$

A bridging function is introduced to apply the missing attributes ϵ_{gh} and by that create the previously missing channel, if possible. There is a known set of bridgeable attributes A_δ. A_δ is set a priori. Virtualised functionalities provide bridges on top of existing sub-infrastructure resources, and they are dependent on the capabilities of the infrastructure. The bridging function applies the missing attribute if $\epsilon_{gh} \subseteq A_\delta$. This supports the data movement and ensures compatibility and interoperability within collaborating nodes. The bridging function δ works as follows:

$$\delta(C) = B, \tag{4.12}$$

Where δ takes as an input the matrix of channels, and returns matrix B of bridged channels. B is represented as a $s \times s$ square matrix. A single entry b_{ij} is a Boolean value, where 0 can mean either a none bridgeable channel or $\epsilon_{ij} = \phi$ (the channel already exists), and 1 means a bridgeable channel c_{ij}.

In case an entry $c_{ij} = 0$, and $\epsilon_{ij} \subseteq A_\delta$, this means that $\delta(c_{ij}) = b_{ij} = 1$. The two concerns that arise dealing with bridging functions are as follows:

- The cost associated with the bridging function offered in terms of time and complexity,
- There exists an infeasibility ratio that needs to be considered in case $\epsilon_{ij} \not\subset A_\delta$; an unsupported data movement is unbridgeable.

The three matrices serve as dependable variables that control the information flow. The aggregation of the channels, rules, and bridgeable channels results in the Information Flow Control (IFC). IFC is calculated by Eq. (4.13):

$$IFC = (R \wedge C) \vee B. \tag{4.13}$$

IFC is a square matrix of the same dimensions $s \times s$. A single entry f is a Boolean value where 0, 1 represent no flow/flow of information, respectively, where $f_{ij} = (r_{ij} \wedge c_{ij}) \vee b_{ij}$. IFC is the adjacency matrix that describes the aggregated directed graph of all possible archetypes in an application scenario. A single application request Q is met when Q and IFC overlap, hence the archetype mapping exists. All matrices we mentioned C, R, B, IFC, Q have the same size $s \times s$.

4.3.7 Application Requests

Once a sub-infrastructure is initialised, application request Q utilises the built framework with collaboration instances. An application request is the matrix of requested data movements among nodes in an application scenario sub-infrastructure.

A single entry q_{ij} is a Boolean value that describes a single requested data flow from n_i to n_j, such that 0, 1 maps to an absence or existence of a requested directional flow. To satisfy an application request Q, Q should overlap with the matrix of supported, allowed, and bridgeable flows IFC.

The previously discussed functions interact as shown in Figure 4.2. N_{App} and μ are the input to create the sub-infrastructure's EPI areas Θ_{App}. Next, the output is used to translate the channels into a matrix using β. Meanwhile, the requirements turned rules are also translated into a matrix of allowed/denied flows by γ. Rules $\wedge$ channels output true in case of alignment, else false. In case of a false, the bridging function is introduced to apply ϵ. After that, the IFC matrix is determined by the aligned and bridged channels, where $IFC = (R \wedge C) \vee B$. A set of Qs is then considered by applying the "XNOR" logic gate with IFC to

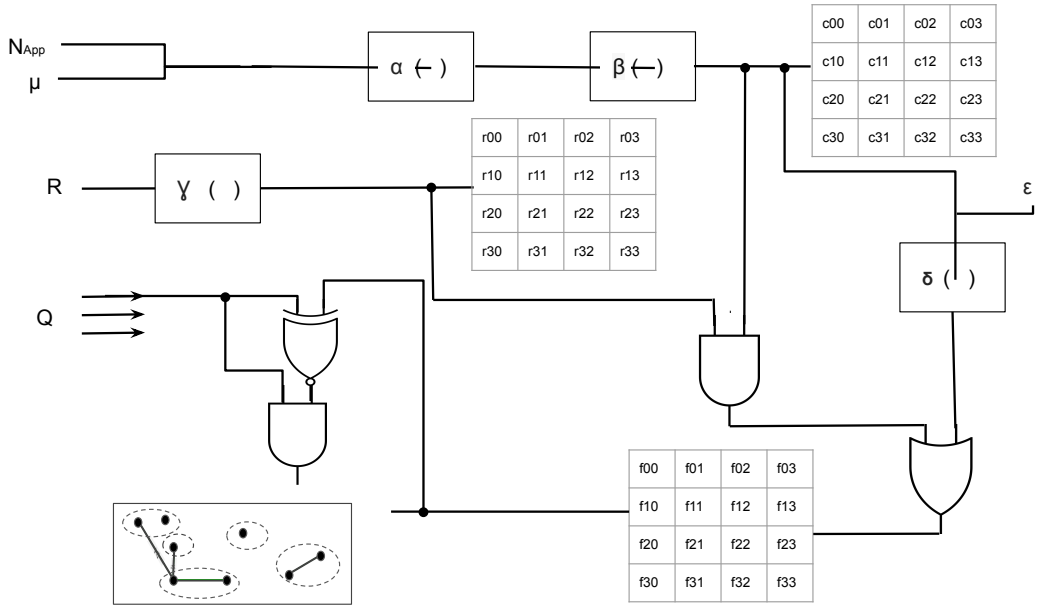


Figure 4.2: The different functions in the EPI infrastructure orchestrator and their relation with the application request.

determine the matches between the two matrices. The "XNOR" output is 1 in case of an overlap between q_{ij} and f_{ij}, and 0 is a mismatch. After that, the logic "AND" is applied to apply (1) or reject (0) a single data movement q_{ij} in an application request Q. That is further clarified in the truth table below, where $q_{ij} = 1$ is applied only when it overlaps with $f_{ij} = 1$, otherwise, the request is rejected.

f	q	overlap	apply
0	0	1	0
0	1	0	0
1	0	0	0
1	1	1	1

4.4 EPI Aggregation Algorithm

After discussing the architecture in section 4.3, we simulate the logic model by running the EPI algorithm. The algorithm works by creating different areas associated with distinct attribute sets, such that N_{App} is an input. It appends nodes with the exact matching attribute set to the same area, as shown in Algorithm 1.

The first lines (3-4) are looping over all the nodes in the N_{App} set and querying the node's attribute set. There are two cases that this algorithm deals with. Lines

Algorithm 1 Algorithm to assign nodes to areas

```
1:  procedure CREATEAREAS(N_App)
2:      Θ ⟵ φ                                   ▷ initialise the set of areas
3:      for all n ∈ N_App do                     ▷ loop over all nodes
4:          A_n ⟵ getAttributes(n)
5:          flag ⟵ 0
6:          for all θ ∈ Θ do
7:              θ ⟵ getAttributes(θ)             ▷ retrieve attributes
8:              if A_n = A_θ then
9:                  θ ⟵ θ ∪ {n}
10:                 flag ⟵ 1
11:                 break
12:             end if
13:         end for
14:         if flag = 0 then
15:             θ ⟵ {n}                          ▷ create new area with node n
16:             Θ ⟵ Θ ∪ {θ}
17:         end if
18:     end for
19:     return Θ
20: end procedure
```

8-11 check whether the area that is associated with this node's attribute set is already there, then it appends it to that area's node set. In the second case coded in lines 14-16, the node belongs to a new area that doesn't already exist.

Once all the nodes are assigned to areas, the algorithm also searches for subset relations between areas (included or distinct), as shown in Algorithm 2. With subsets initialised it is easier to search for supported channels between nodes within the areas. This relation is determined according to the attribute set associated with each area.

In Algorithm 2, we loop over created areas twice (lines 1-3) to compare the two area's associated attribute sets. The algorithm sets a subset-superset relation between areas (lines 7-11).

The model's logic dictates that if an area is a subset (with fewer attributes) of another area, it means that supported channels exist between this area's nodes to the nodes in all superset areas (with more attribute characteristics). Algorithm 3 processes the created areas and deduces channels, as follows.

Algorithm 3 loops over nodes in each area (lines 2-3), and initialises existing channels between nodes within the same area (lines 4-6). In lines 9-12, the algorithm checks whether a node's area has a subset-superset relation area, if so channels between this node and the nodes in the other area are initialised by setting the corresponding entry in matrix C to true.

Algorithm 2 Algorithm to assign subsets to areas

```
1: for all θ_i ∈ Θ do                                   ▷ loop over areas
2:     superset_i ⟵ φ                                   ▷ The set of all supersets to area
3:     holder ⟵ φ
4:     for θ_j ∈ Θ do                                   ▷ compare with other areas
5:         A_i ⟵ getAttributes(θ_i)
6:         A_j ⟵ getAttributes(θ_j)
7:         if A_i ⊂ A_j and i ≠ j then                  ▷ i is a subset to j
8:             holder ⟵ holder ∪ {θ_j}
9:         end if
10:    end for
11:    superset_i ⟵ superset_i ∪ {holder}
12: end for
```

Algorithm 3 Algorithm to deduce channels between nodes

```
1: procedure checkChannels(Θ)
2:     Let C = {c_ij} be a new sxs matrix with ∀c_ij = 0
3:     for θ_i ∈ Θ do                                   ▷ loop over areas
4:         for n_i ∈ θ_i do                             ▷ loop over nodes in each area
5:             for n_j ∈ θ_i do                         ▷ nodes in the same area
6:                 if i ≠ j then
7:                     c_ij ⟵ 1                         ▷ channel exists
8:                 end if
9:             end for
10:            if supersets_i ∈ φ then                  ▷ θ_i has supersets
11:                for θ_j ∈ supersets_i do
12:                    for n_z ∈ θ_j do
13:                        c_iz ⟵ 1
14:                    end for
15:                end for
16:            end if
17:        end for
18:    end for
19:    return C                                         ▷ the matrix of all channels
20: end procedure
```

After creating areas and deducing their channels, we consider the rules. We assume that the rule matrix was set by the policy management system involved, and then translated with γ. Then, take into account the possible bridging functions that can compensate for missing attributes and add to supported channels C. Function $\delta(C)$ results with the matrix B, where $bij = 0$ in case channel already exists $\epsilon_{ij} = \phi$ or isn't satisfied by A_δ.

The algorithm checking for available bridges (matrix B) and matching it to the missing channels between nodes is as in Algorithm 4.

Algorithm 4 Algorithm to add bridges

```
1:  procedure CHECKBRIDGES(C)
2:      for n_i ∈ N_App do
3:          A_i ⟵ getAttributes(n_i)
4:          for n_j ∈ N_App do
5:              A_j ⟵ getAttributes(n_j)
6:              if c_ij = 1 then
7:                  Channel already exists
8:              else
9:                  ϵ_ij ⟵ A_j ∩ ¬A_i                ▷ the missing attributes
10:                 A_δ ⟵ getBridges()                ▷ retrieve bridges
11:                 if ϵ_ij ∈ A_δ then                ▷ bridgeable attributes
12:                     Apply bridging functions
13:                     c_ij ⟵ 1
14:                 end if
15:             end if
16:         end for
17:     end for
18:     return C                                      ▷ updated channels
19: end procedure
```

In Algorithm 4, the algorithm in line 2 loops over all nodes in N_{App} and checks if it has supported channels to other nodes in lines 6-7. In case the channels to other nodes are missing, the algorithm in lines 11-13 checks if the attribute difference between the two nodes is bridgeable. As a result, the algorithm returns the matrix of updated channels with the added bridges.

The IFC matrix is pushed into the sub-infrastructure to set it up for possible application requests. Application requests are successful when Q overlaps with IFC. Otherwise, not all data movements within this request might be possible. We deploy the algorithm on the server to emulate the orchestrator prototype.

4.5 Evaluation

In this section, we introduce an orchestrator prototype (Figure 4.3) that uses the above methodology, and we use that to measure the algorithm's performance. Moreover, we propose the formulas to evaluate the infrastructural properties that affect the feasibility and cost of different application requests.

4.5.1 Performance

We set up a server to run the algorithm in the back-end and maintain a database of all possible nodes. Then we initiate an application scenario with the specifying nodes relevant for this request N_{App} (step 1 in Figure 4.3). The framework redirects the application query to the policy management system to query the regulating rules R of the information flow within this scenario (steps 1.1 and 1.2).

The server runs the logic model and lists the areas Θ_{App}, estimate B, and aggregates it with R (step 2). IFC is deduced with all possible archetype models, and the setup cost is estimated (step 3). After the confirmation of the user (parties initiating the application) (step 4), the IFC setup is pushed to the infrastructural level (step 5). Ultimately, application requests (Q) are then initiated to utilise the sub-infrastructure (step 6), and requests are applied or rejected accordingly.

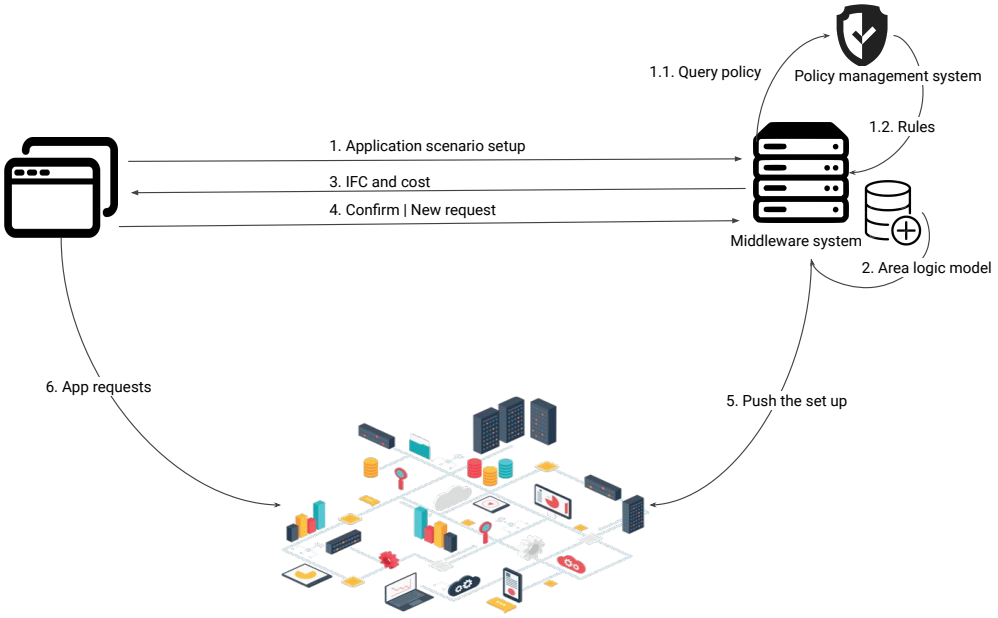


Figure 4.3: The EPI infrastructure orchestrator prototype and an illustration of the operational workflow interacting with the middleware system.

We further evaluate step 2 in Figure 4.3 that runs the area logic model algorithm, and we estimate the performance of the infrastructure orchestrator (Algorithms 1, 2, 3, and 4). This measurement is important to test how would the algorithm scale with large input nodes in case of time-critical application requests. The objective is to measure the performance as the number of nodes increases and becomes more distinct (heterogeneous attribute sets).

The performance variables that we measure are execution time and CPU time. The execution time is measured by the elapsed real-time between the invocation of the script and its termination, and the CPU time is measured by the processing

time of the instructions. This describes the scaling of the delay between sending an application scenario setup request (step 1) and receiving an answer to the IFC matrix (step 4), which is relevant in the case of time-critical requests.

The algorithm's complexity also determines the performance of the orchestrator. The plot in Figure 4.4 shows the scaling of response time in real-time and CPU time. The time is measured as the input size increases (number of nodes increases) and as the heterogeneity of the input increases (number of areas increases). In the following plot, we keep the heterogeneity as a constant ($AHR = 0$), and we increase the number of nodes. Notice that a third-degree polynomial nicely fits the data, demonstrating time complexity $O(n^3)$. Figure 4.4 provides proof of our algorithm scales, where the time recorded follows the $O(n^3)$ with negligible errors.

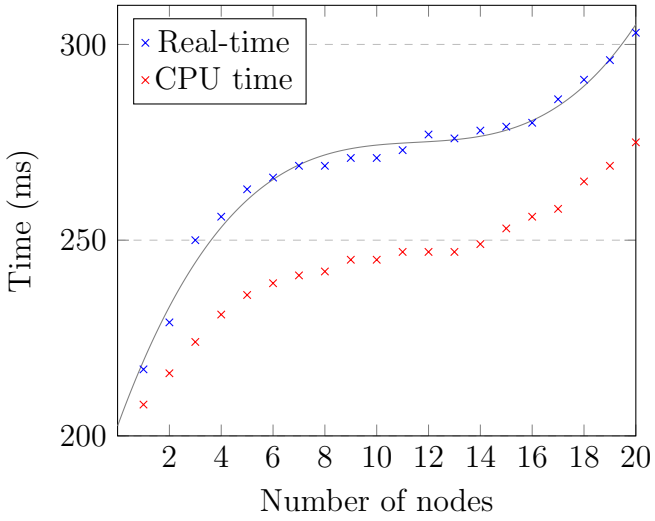


Figure 4.4: Script execution time depending on increasing input

To calculate the average intersection rate AIR for μ, we consider the intersection rate of all distinct pairs in μ, as in Eq. (4.14).

$$AIR = \frac{\sum_{i,j;i<j}^{n} \frac{||A_i \cap A_j||}{||A_i \cup A_j||}}{\binom{n}{2}} \tag{4.14}$$

To determine the average heterogeneity rate AHR of the nodes' attribute sets, we calculate AHR, such that:

$$AHR = 1 - AIR \tag{4.15}$$

The plot in Figure 4.5 shows the scalability of real and CPU time as AHR increases, where $n = 20$. Notice that a second-degree polynomial nicely fits the scatter plot, indicating a scaling as $O(n^2)$ with negligible error margin. The plot

trend is explained by the complexity of the implemented algorithm due to the number of loops needed.

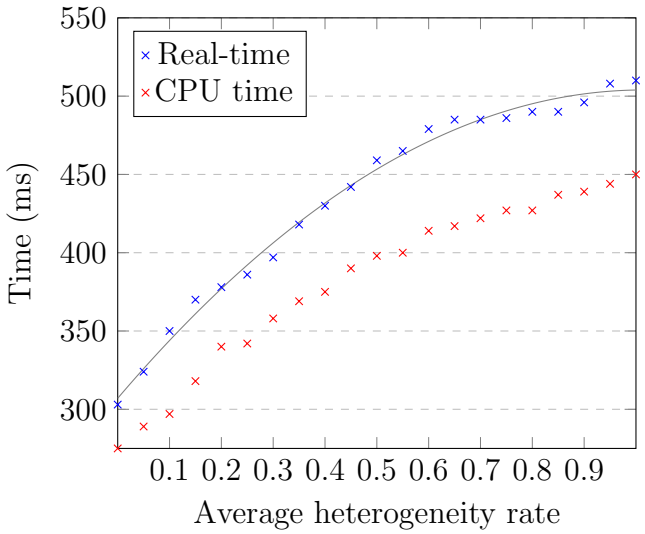


Figure 4.5: Script execution time depending on increasing AHR

4.5.2 Infrastructural Properties

The percentage of aggregated nodes within the same area indicates the setup cost of any sub-infrastructure. As an example, if the aggregation power percentage is 100%, that means that no extra bridging functions need to be in place before the movement of information between nodes is possible. The opposite is also true, the lower the aggregation power, the higher the setup cost might be estimated. Moreover, aggregation is directly proportional to the probability of satisfiable application requests. As an example, if the aggregation power is 100% (all areas on 1 node) then the probability that supported channels exist (data movement is possible) is 1.

Eq. (4.16) calculates the number of possible attribute subsets, given that d is the number of all possible distinct attributes.

$$||\wp(A)|| = 2^d - 1 \tag{4.16}$$

The result of Eq. (4.17) shows the percentage of nodes being aggregation into distinct areas. The value might differ with different infrastructures. Aggregation power is calculated as follows, where Θ is the set of all areas grouping N.

$$Aggregation = \frac{||\wp(A)|| - ||\Theta|| + 1}{||\wp(A)||} \times 100 \tag{4.17}$$

The infrastructure also has a property of setup success ratio, where the bridging functions can satisfy all possible missing channels, hence attributes ϵ_{ij}. The fewer attributes that are bridgeable, the higher the failure ratio is. The success ratio is calculated as follows:

$$Success = \frac{||A_\delta||}{||A||} \tag{4.18}$$

4.6 Discussion

The orchestrator provides a clear identification of missing attributes per node and matches it to a bridging function. The available bridging functions directly affect the application request feasibility and success rate, as defined by the infrastructural properties. The infrastructural properties rely on the infrastructure itself, and to accomplish a complete success rate, the infrastructure should be ready to bridge any missing attribute. Moreover, the success rate also depends on the heterogeneity of the infrastructure, which the aggregation power value reflects. The aggregation power is 100% if the areas created are equal to 1 (all nodes are characterised by the same attribute sets). Subsequently, upgrading the infrastructure improves the success rate of an application request and the aggregation power of areas.

The algorithm of the logic model scales under the complexity of $O(n^3)$. The algorithm performance is interpreted as a one-time run delay before pushing the setup instructions into the infrastructure level. After that, the complexity and time cost are inversely proportionate to the aggregation power and average intersection rate. In other words, as the heterogeneity in the infrastructure increases, the time cost increases. That can be explained by the extra steps of intermediate bridging functions that are associated with higher heterogeneity and a larger number of areas.

The previously discussed measurements foresee the probability of an application request failure, high setup costs, and delays. Those values are dependent on the infrastructure and can be improved by tailoring the infrastructure resources. The properties can be evaluated before introducing any application to estimate a future upgrade or to customise the requests accordingly.

Existing health data sharing frameworks focus on primarily security and access control, as shown in Table 4.1. The two features can be addressed within the EPI framework by considering security and access control attributes, hence channels exist to support data movements to nodes that offer the same security functionalities (same area) or more (area superset). Our proposed framework addresses other features by defining attributes (for example: structured data storage) that ensure interoperability, and delegating the definition of data-sharing rules to the policy management system. Data auditing is also a proposed feature considered by the EPI architecture, as shown in Figure 4.1. The EPI infrastructure orches-

trator adds the logic model to the infrastructural programmability and provides a dynamic and heterogeneous infrastructure. As a result, we contribute to existing frameworks by offering the dynamicity feature.

4.7 EPI use cases

In Figure 4.6, we show an example network topology with the middleware controller running on top. At the bottom layer, we have the physical topology of the network nodes. The EPI infrastructure orchestrator creates the logical topology by grouping the nodes into areas. To communicate between areas, the middleware set up the proxy nodes with Area-Area proxy commands (specified by δ). The proxy will connect the node to the NFVI and bridge the incompatible attributes. Note that in the following use cases, the attributes that we considered are charac-

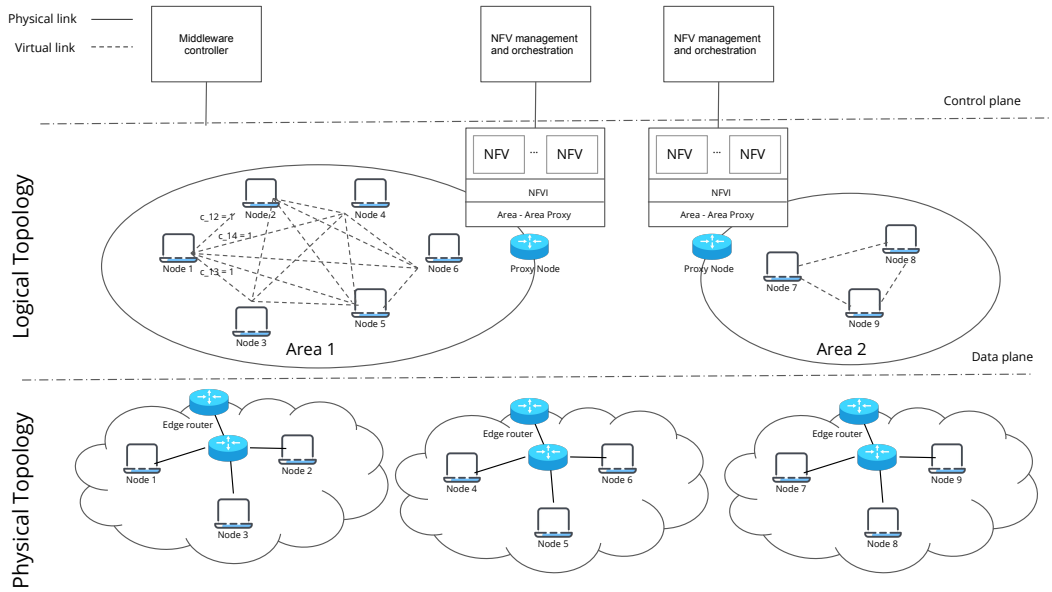


Figure 4.6: The network topology in an EPI use case.

teristic of the nodes' low-level communication attributes, such as the IP version and data encryption. In this section, we introduce these use cases, apply the logic model to determine the IFC, and set up the bridging functions proxies in an attempt to satisfy the requested data movements. Each node is identified as follows $n_i(A_i, IP)$, and the possible attributes are the set A = {Encrypted data (a_1), IPv4 (a_2), IPv6 (a_3)}. If a node has a_1 as an attribute, that means that this node store, maintain, and communicates the data in an encrypted form. Attributes a_1 and a_2 mean that this node supports IPv4 and IPv6 addressing, respectively. Any incompatibilities between these attributes can result in an unsupported data movement, hence data sharing failure.

The first use case is proposed to share an ML model and train it over different nodes. That is described in Figure 4.7. n_1 is the ML model owner and wants to train the model on the different medical data sets stored on n_2, n_3, and n_4 and representing potential hospital parties. Figure 4.7 shows n_1 initiating the application scenario setup and identifying relevant nodes for this application. In this case, $N_{App} = \{n_1, n_2, n_3, n_4\}$, and $A_1 = \{a_2\}$ $A_2 = \{a_1, a_3\}$ $A_3 = \{a_1, a_3\}$ $A_4 = \{a_1, a_2\}$ respectively. The middleware system runs the logic model, assigns areas based on these nodes' attributes, deduces IFC and sets up the area-area proxies. We assume that the policy allows all data movements and the R matrix

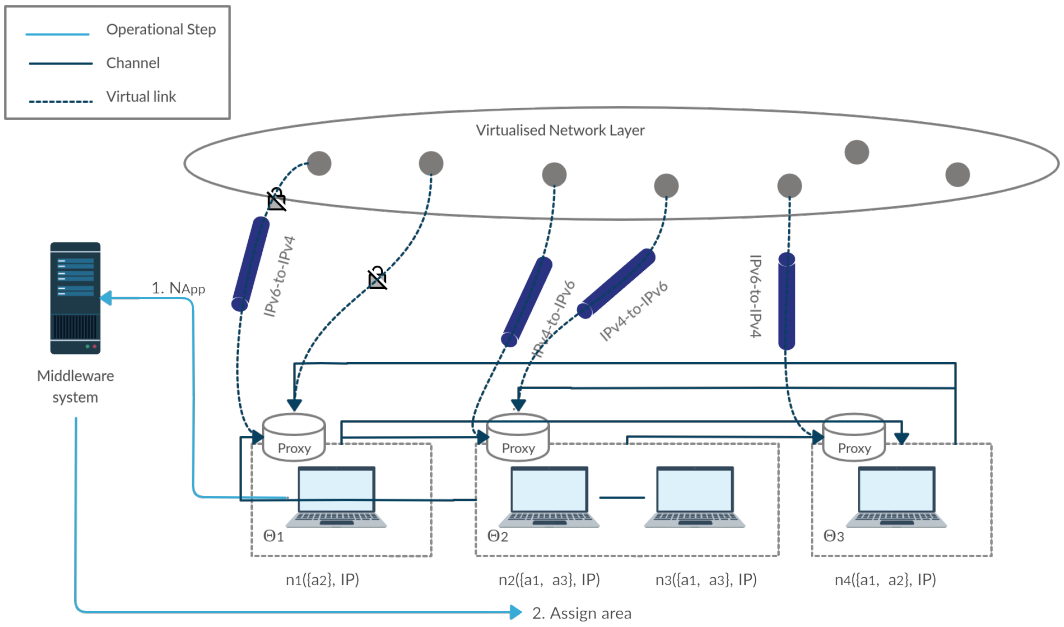


Figure 4.7: The EPI set up for the ML model sharing use case.

entries are initialised all to 1. The bridging functions utilised are the IPv4-to-IPv6 tunnelling and data decryption functions. In case of an unsupported movement between nodes, the middleware system sets up a proxy that connects to the NFVI through a virtual link to apply the bridging functions.

The proxy's functionalities (δ) can be shown in this use case in Figure 4.7. As an example, with the incompatibility between θ_1 and θ_2, $\epsilon_{12} = \{a_1, a_3\}$. The proxy manages bridging previously unsupported channel with $\delta(c_{12}) = decrypt \circ tunnel(c_{12})$. After applying these functions, the node can successfully receive and process the data communicated.

Figure 4.7 shows the supported channels, checks bridgeable channels, and

calculates IFC as follows:

$$IFC = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \wedge \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \vee \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Data movement requests Q is applied once it overlaps with the IFC matrix. That is determined as follows, $\otimes$ operates as a logic XOR:

$$\neg \left(IFC \otimes \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \right) \wedge \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

The second EPI use case is maintaining an EHR storage node with multiple data sources. In this case, we have data sources of doctors running on n_1 and n_2 nodes. The EHR storage is maintained on n_3, and n_4 acts as a backup. n_1 and n_2 want to have a full story of the patient's medical records on n_3, and n_3 will update n_4 accordingly. In this case, $N_{App} = \{n_1, n_2, n_3, n_4\}$, and $A_1 = \{a_1, a_2\}$ $A_2 = \{a_1, a_3\}$ $A_3 = \{a_1, a_2, a_3\}$ $A_4 = \{a_1, a_2, a_3\}$ respectively. Figure 4.8 illustrates this use case and sub-infrastructure setup. In this case, the encryption/decryption of data is supported on all nodes. Subsequently, the bridging functions that are managed by the proxy are IPv4-to-IPv6/IPv6-to-IPv4 tunnelling to support data movements between these incompatible nodes. As an example, $c_{12} = 0$ indicate unsupported data movement, then the proxy performs $\delta(c_{12}) = tunnel(c_{12}) = b_{12} = 1$.

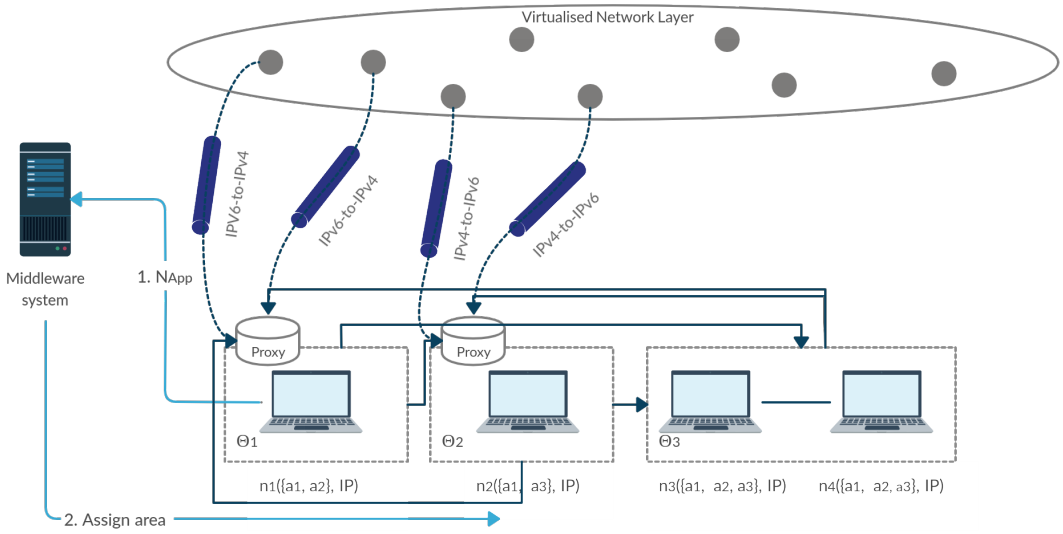


Figure 4.8: The EPI set up for EHR storage and backup use case.

Figure 4.8 shows the supported channels, assumes all rules' entries are set to 1, checks bridgeable channels, and calculates IFC as follows:

$$IFC = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \wedge \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \right) \vee \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Data movement requests Q is applied once it overlaps with the IFC matrix. That is determined as follows:

$$\neg \left(IFC \otimes \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right) \wedge \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The third EPI use case is patient data streaming and personalised diagnosis. Nodes 1, 2, and 3 simulate the behaviour of patients running on n_1, n_2, and n_3. n_3 has a trained model and aims to receive the monitoring data and reply with a diagnosis. In this case, $N_{App} = \{n_1, n_2, n_3, n_4\}$, and, $A_1 = \{a_1, a_4\}$ $A_2 = \{a_1, a_2, a_4\}$ $A_3 = \{a_1, a_2, a_3, a_4\}$ $A_4 = \{a_1, a_2, a_3, a_4\}$ respectively. Figure 9 illustrates this use case and infrastructure setup.

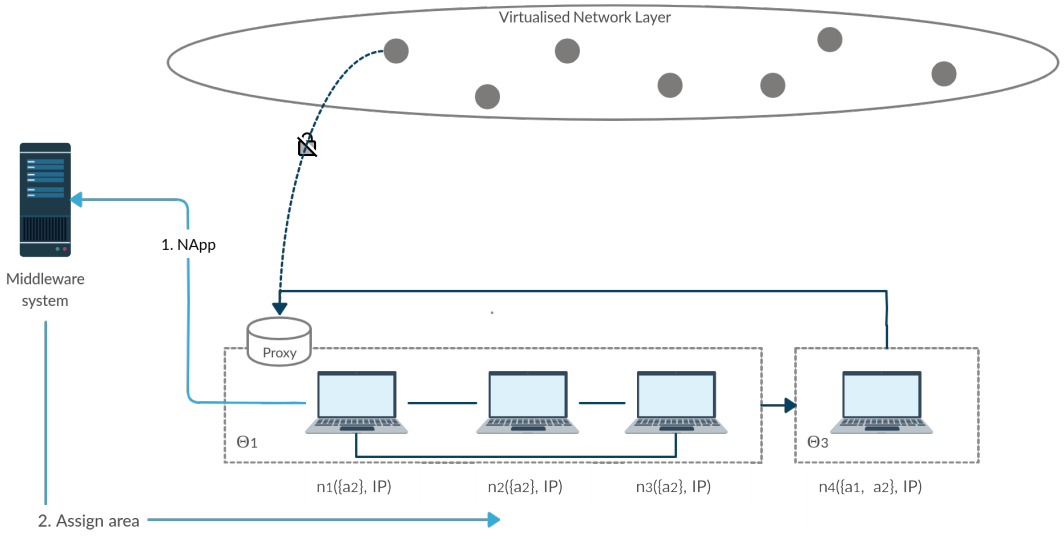


Figure 4.9: The EPI set up for EHR storage and backup use case.

Figure 4.9 shows the supported channels, checks bridgeable channels, and

calculates IFC as follows:

$$IFC = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \wedge \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \vee \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Data movement requests Q is applied once it overlaps with the IFC matrix. That is determined as follows:

$$\neg \left(IFC \otimes \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \right) \wedge \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

As a result, the EPI orchestrator provides the previously missing support to effectively share data between EPI parties in different use case scenarios. It is important to highlight that other attributes might not be bridgeable, where $a \notin A_\delta$. As an example, when we consider more attributes like the TLS version, then abiding with the model's requirement ensures that nodes with only compatible versions can communicate which ensures enhanced security. On the other hand, in the case of TLS version incompatibilities, non-existing channels between nodes are non-bridgeable in practice.

4.8 Conclusion

In this chapter, we defined the logic model and provided the mathematical notations to follow to reason about infrastructure setup automation. Then we simulated the Area logic algorithm and designed a prototype to use it. This approach gives us the guidelines for the framework, which can be used further to run a healthcare application or any other application with sensitive data constraints. The performance measurement formulas of the infrastructure aggregation and success rate are used to define the infrastructural properties, and then estimate failure and cost. We evaluated the performance of the algorithm and estimated that it scales with $O(n^3)$ as an upper limit, which is the time complexity of the operational workflow interacting with the framework. This chapter lays out the infrastructure orchestrator's logic model, and we plan to extend possible Bridging Function (BF) implementations, we implement different traffic proxying methods, and investigate the tradeoffs of each to efficiently bridge a security gap in compliance with governing policies.

Chapter 5

Bridging Function Chains Orchestrator: Feature Implementation

In efforts towards feature implementation of the EPI infrastructure orchestrator, one of the main challenges is to manage and proxy traffic to enforce security and network low-level policies and secure data-sharing. The proposed dynamic orchestration framework defines the topology of the service chains to enforce network and security policies by instantiating Service Function Chains (SFC's) on the fly via lightweight and easily deployable containers.

The design choice of container-based Network Functions (NFs), encapsulated in lightweight Docker containers, ensures platform independence and efficient resource utilization. The chapter evaluates packet redirecting tools, crucial for enforcing policies through Bridging Functions (BFs) containers responsible for network and security services. The investigation includes benchmarking reverse proxy and SOCKS-based implementations, considering latency overhead, processing rate throughput, and defined parameters. The benchmarking results in a comparative analysis with broader applicability. We analyse the results according to the expected traffic movement with each tool, and we base our recommendation on what tool to utilise within the EPI framework under different use case scenarios' performance requirements. This chapter partially answers the question:

RQ2: "What are the performance tradeoffs when employing different packets' redirection methods and bridging functions to enforce network policy-compliant routes under different workloads?"

This chapter is based on:

- **J. A. Kassem**, O. Valkering, A. Belloum and P. Grosso, "EPI Framework: Approach for Traffic Redirection Through Containerised Network Functions," 2021 IEEE 17th International Conference on eScience (eScience), Innsbruck, Austria, 2021, pp. 80-89, doi: 10.1109/eScience51609.2021.00018.

5.1 Introduction

Effectively processing and sharing healthcare data securely remains a complex challenge, prompting the exploration of dynamic network infrastructures within the EPI project. This chapter emphasizes the pivotal role of proxying and re-routing traffic in ensuring secure health data-sharing across diverse domains. By leveraging the principles of Software-defined Infrastructures (SDI) and container-based Network Functions (NFs) encapsulation, the EPI Framework (EPIF) seeks to empower patients through personalised interventions.

Central to our investigation is the deployment of lightweight Docker containers encapsulating NFs, specifically focusing on bridging functions (BFs). These BFs play a critical role in enforcing network and security services by redirecting and controlling traffic. The chapter is dedicated to evaluating various packet redirecting tools and exploring their impact on network performance. Our objective is to understand the trade-offs involved in dynamic traffic re-routing and its influence on infrastructure responsiveness.

In addition to addressing the technical intricacies of proxy implementations, the chapter introduces a novel data-sharing framework supporting healthcare applications. A significant aspect of our contribution lies in the benchmarking of different redirection tools, providing insights into their performance metrics, overhead, and processing rates. By examining the implications of traffic re-routing on network performance, we aim to enhance the understanding of these mechanisms for secure healthcare data-sharing.

In this paper, we evaluate packet redirecting tools that would serve as a building block of the proposed EPI framework - EPIF ([2]). We implement several methods to effectively control traffic routing and redirection. That will be used to force traffic through Bridging Functions (BF's) containers (network and security services).

At a lower level, we need appropriate technologies to implement efficient traffic redirection functionality. We expect that the infrastructure's dynamicity comes at a price of network performance and overhead. We aim to experiment and investigate the impact of introducing different redirection proxies in the middle of a data-sharing session. Note that, aside from data records, sharing algorithms

and sharing processed data are also considered.

This chapter is organized as follows: In Sec. 8.8, we introduce the EPIF in the context of personalized medicine. In Sec. 5.2, we elaborate on the concept of bridging functions and the area logic model employed in the EPIF. In Sec. 5.3, we touch on the implementation designs of the proxy component utilizing a traffic redirection functionality. In Sec. 5.4, we illustrate the protocol traffic of each proxy in order to estimate the connection setup time of each. In Sec. 5.5 we detail the evaluation of the implementations with some insights. In Sec. 5.6, we compare the proxies according to a number of parameters. Lastly, we showcase related work in Sec. 8.2 and conclude our work and introduce potential future work in Sec. 5.8.

5.1.1 The EPI framework

EPIF is a dynamic health data sharing framework that accommodates different domains' infrastructural capabilities by shipping and managing containerised security and network services called *bridging functions*, which run at special *proxy nodes*. We will describe in detail these capabilities in Sec. 5.2. An automated setup of the infrastructure is required to achieve:

- Reachability of the end-points nodes;
- Optimal security across collaborating domains;
- Reasonable network performance;
- Bridging services availability;
- Hardware selection and scalability (horizontal vs vertical scaling);
- At a higher level, abiding and enforcing policy management requirements.

As shown in Fig.5.1, EPIF has different components that are crucial to automate the data-sharing processes between participating parties. The main components of the framework are the orchestrators (both at the application level and infrastructural level); the policy management system and the components required to be present at the participating institutions, namely the resource provisioner and the authorizers.

5.1.1.1 Orchestrators

EPIF employs a dual orchestrator design. There is an application-focused orchestrator and an infrastructure-focused orchestrator. The two orchestrators, in principle, operate independently but exchange runtime information so they can collaboratively enforce policies and give insight into the functioning of the combined deployment.

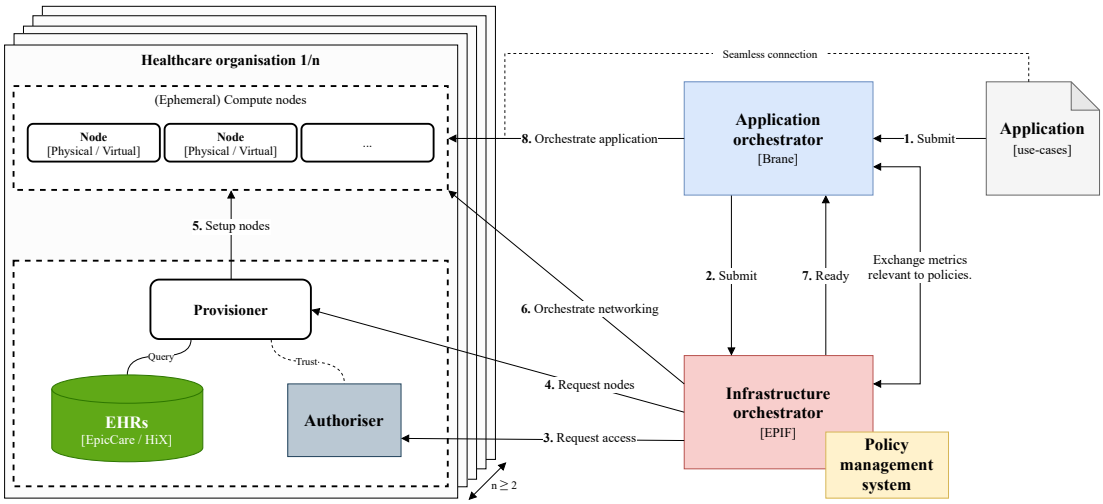


Figure 5.1: A high-level view of the different EPIF components and their interactions after an application request.

- **The application orchestrator** is the EPIF starting point for executing applications. After receiving an application submission (step 1 - Submit - in Fig.5.1), it first sends a request to the infrastructure orchestrator (step 2 - Submit) to streamline any discrepancies between the targeted infrastructure domains. This streamlining, described in detail in Sec. 5.2, is done transparently to the application orchestrator. The final result of the communication with the infrastructural orchestrator is the receipt of confirmation of setup (step 7 - Ready). At this point, the application orchestrator considers the targeted infrastructure domains as ready and that there is an interconnected execution environment. The application orchestrator will handle the heterogeneity of the made-available resources, i.e. compute nodes by direct interactions (step 8 - Orchestrate Application).

- **The infrastructure orchestrator** arranges all aspects related to the multi-domain target environment. The orchestrator requests access to the domain resources (step 3 - Request Access); after being granted access it requests specific nodes (step 4 - Request Nodes); finally, it manages the required bridging functions (BF) that are needed based on the area logic model (Sec. 5.2.1) via step 6 - Orchestrate Networking.

Decoupling the two orchestration concerns into separate systems increases the applicability of the EPIF. That is apparent with the infrastructure orchestrator's independent operation that makes it possible to support alternative application orchestrators, such as Vantage6 (84) which is a privacy-preserving federated learning infrastructure. On the other hand, implementers targeting the EPIF can deploy the application orchestrator independently for development and testing

purposes to improve productivity.

Brane ([106]), a framework for programmable orchestration, is at the core of both orchestrators. Programmability is an essential aspect of enabling dynamic adaption to heterogeneity across infrastructure domains.

5.1.1.2 Policy management system

The policy management system captures allowed and denied data flows. It considers different constraints such as GDPR, patients' consent, and the intended goal of the application scenario. The system can deduce extra setup requirements and communicate them to the infrastructure orchestrator via specific Domain-Specific Languages. The communicated policy is estimated to be dynamic, as it can change during an application request, and the EPIF is asked to adapt and comply with any policy change.

5.1.1.3 Domain components

At all the participating organizations a resource provisioner and authoriser components are required. They might be implemented directly by the domain, in which case they have to implement the proper APIs to interact with the EPIF, or they can be provided as pluggable components from the EPIF itself, in which case they are compatible with the infrastructure orchestrator.

In the specific case of one single domain, an EPI client can run an application on (physical/virtual) node(s) within one healthcare domain by only submitting a request via Brane. This is illustrated in Fig. 5.1 with steps 1 and 8. In all other cases, we need multi-domain communication and this is taken care of by the EPIF components in steps 2-7.

5.2 Bridging policies and infrastructural resources

A core element of the EPIF operations is the need to match the policy constraints provided by the policy management system with the underlying capabilities of the infrastructure. To support this, we defined in ([2]) two core concepts: *security areas* and *bridging functions*.

The area logic model works on grouping end-point nodes partaking in a data-sharing application according to the associated network and security functions that are supported and applied by the node itself into security areas. After that, the logic model deduces which bridging functions need to be instantiated to move data between nodes in different areas.

5.2.1 Area logic model

A health domain party has a number of nodes which are the endpoints of physical/virtual resources. We represent all resource endpoints included in the infrastructure by the set N of nodes:

$$N = \{n_i \mid i = 1, ..., m\}, \tag{5.1}$$

where n_i represents a single node in the infrastructure and m is the total number of nodes.

To evaluate the security and network capabilities of a node with a domain, we consider the associated attributes set. Attributes refer to the node's supported and applied network and security functionality within a network, such as segmentation, data encryption and key management, firewalls, scalable network links, etc. Let A be the set of all possible attributes:

$$A = \{a_k \mid k = 1, ..., d\}, \tag{5.2}$$

where a_k represents a distinct attribute and d is the total number of attributes.

Once the application scenario is made clear, the infrastructure is initialised to support it. Let N_{App} be the set of nodes relevant to an application scenario, such that:

$$N_{App} \subseteq N, \tag{5.3}$$

where N is the silo of resources (physical/virtual nodes) of all parties and N_{App} specifies the nodes collaborating in a specific application scenario. Each node's capabilities are described by a set of attributes, which can be any subset of A. Subsequently, every node n_j in N_{App} is assigned $A_j \subseteq A$.

After we define N_{App} and its associated attributes, we can evaluate these nodes' security and reachability across domains. The area abstraction is used to identify heterogeneity between nodes and deduce which data movements are supported and feasible: we call such supported communication between nodes in different areas *channels*. The logic dictates that data movement is supported when Eq. (7.4) is satisfied, where $n_g \Rightarrow n_h$ represents a one data movement support between nodes n_g and n_h. That means that a directional movement from n_g to n_h is supported when the capabilities of n_h (A_h) is the same or a superset of that of n_g (A_g)

$$n_g \Rightarrow n_h, iff A_g \subseteq A_h. \tag{5.4}$$

5.2.2 Bridge attribute gaps

Application requests, policy rules, and available channels might not necessarily be compatible, and this can be an issue to run an application successfully. Continuing with our formalism, we can see that condition $A_g \subseteq A_h$ supports data movement

from node n_g to n_h. Otherwise, A_g is not a subset of A_h and communication is in principle not possible.

As shown in Fig. 5.2, BF's are introduced to bridge attribute gaps and subsequently enable communication that wasn't previously supported across different areas and domains. Concretely, a bridging function is introduced to apply the missing attributes ϵ_{gh} and by that enable the previously missing communication, if possible. In Eq. (5.5), ϵ_{gh} indicates the missing attributes

$$\epsilon_{gh} = A_h \cap \overline{A_g}. \tag{5.5}$$

There is a known set of bridgeable attributes A_δ. A_δ is set a priori. A_δ is bridged by associated BF's that are containerised and shipped to be pooled on the proxy nodes. BF's are independent of the capabilities of the infrastructure. The bridging function applies the missing attribute if $\epsilon_{gh} \subseteq A_\delta$. This supports the data movement and ensures reachability and security within collaborating nodes across different domains.

The BF's images are provisioned and maintained in proxy nodes. As illustrated in Fig. 5.3, there exists a proxy per network, which allows the proxy to act as a single mini-orchestrator. The proxy intercepts, controls, and manipulates traffic. The EPIF employs redirection tools to enforce traffic through the proxy and consequently through the BF. Note that it is sometimes needed to have a number of cascading bridging functions, ie *function chains*, to put two areas in communication.

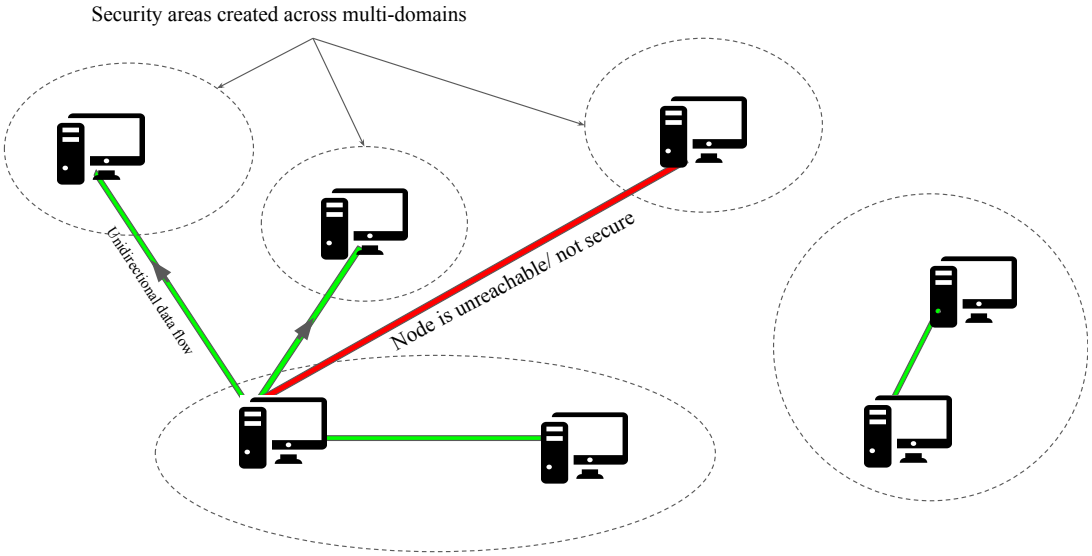
The architecture in Fig. 5.3 shows how the proxy components fit with previously defined components in Sec. 8.8.

Three main concerns arise when dealing with bridging functions:

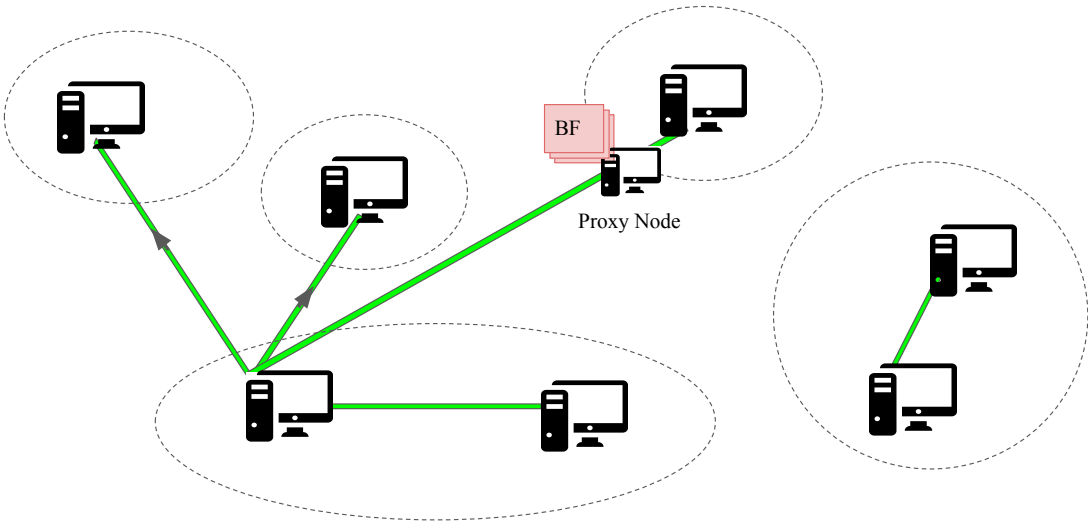
- *infeasible bridges:* There exists an unfeasibility ratio that needs to be considered in case ϵ_{ij} *not* $\subset A_\delta$; an unsupported data movement is unbridgeable.
- *costs:* There is a cost associated with the bridging function in terms of time and complexity to set it up;
- *redirection tools:* There is the need to identify and test the appropriate tools to effectively intercept and redirect traffic through the instantiated BF's running on the proxy.

The EPIF manufactures by software network and security container-based functionalities, and host it on the proxy node. The newly introduced functionalities are flexibly and dynamically traded and provisioned by the orchestrators. As a result, the EPIF adapts the underlying infrastructure to support health applications (e.g. medical data streaming, EHR and backup, machine learning model training) with different archetypes.

In the next sections, we address the last concern and introduce the different proxy implementation approaches and the performance of the redirection tools.



(a) Initial unsupported data flow in multi-domain environment



(b) Enabling previously unsupported channel via instantiating BF's at a proxy node

Figure 5.2: An illustration of an example scenario of different domain nodes belonging to different security areas

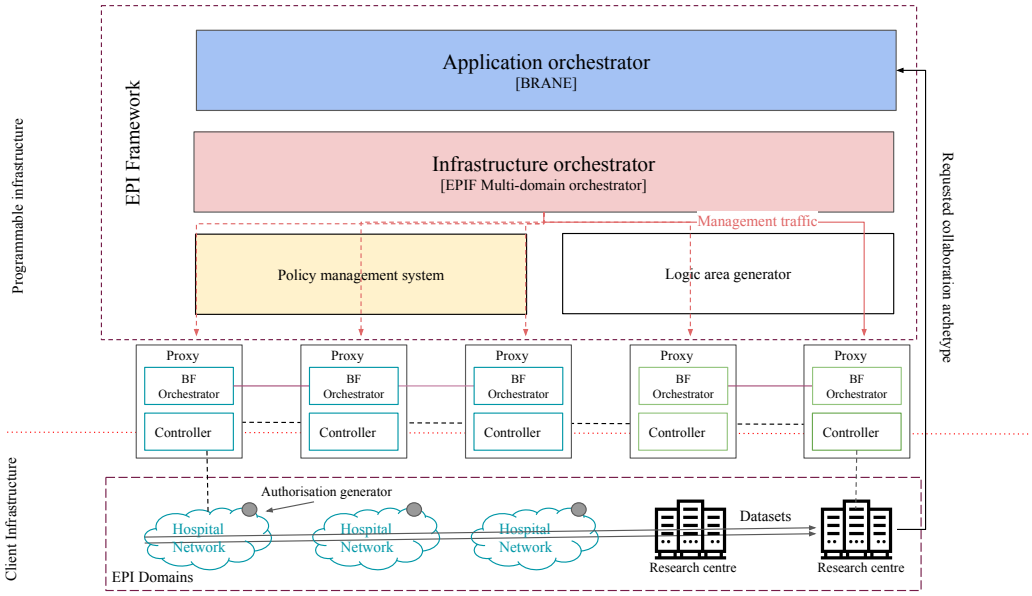


Figure 5.3: A figure showing the EPIF architecture with different EPIF components running (including the proxy node).

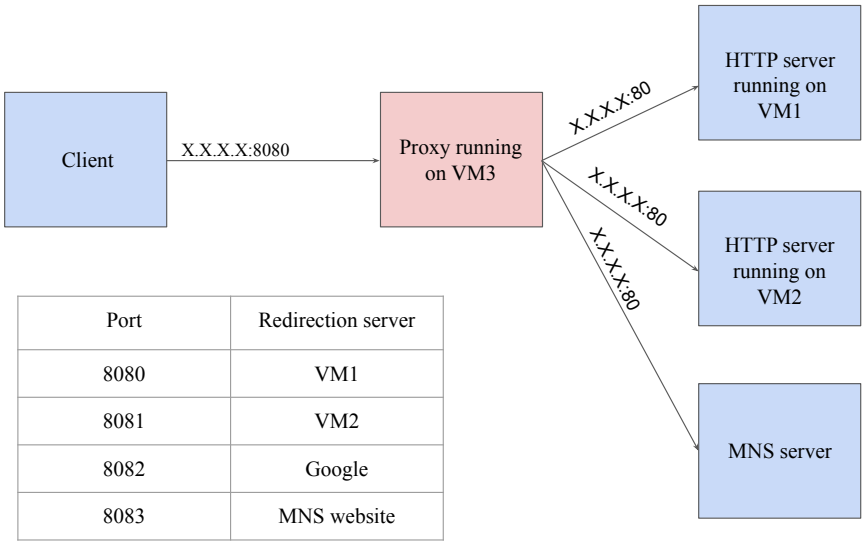
5.3 Proxy implementations

We evaluate two proxy implementation candidates for the EPIF. The first candidate that we consider is a reverse proxy. The second candidate is a SOCKS-compatible proxy. Nodes route outgoing network traffic through a separate proxy node. On the proxy node, the appropriate BF chain processes the network traffic before the proxy implementation sends the network traffic to the destination.

5.3.1 Reverse proxy

The reverse proxy approach implements the NGINX reverse proxy tool (9). This proxy tool is widely used to balance load through multiple servers, display content from multiple websites in a seamless manner, or transfer requests for handling to application servers using protocols other than HTTP. The reverse proxy works by proxying a request, redirecting it to the specified server, retrieving an answer, and sending back the reply to the client. The proxy acts as the middleman handling requests, redirecting them, and forwarding back the reply. The proxy can handle requests to non-HTTP servers (for example, PHP or Python) using a specified protocol. The implementation of the reverse proxy is illustrated in Fig. 5.4. The redirection table is customised and initialised at the proxy VM, and each port is coupled to identify a different destination server. This approach is very simple and effective, but it is vital to know the exact port to reach beforehand. Moreover, several reverse proxies can be required to handle BF chaining, however, the route

must be static or reconfiguration is needed.



Port	Redirection server
8080	VM1
8081	VM2
8082	Google
8083	MNS website

Figure 5.4: Reverse proxy implementation, showing the redirection mechanism for data flow redirection through the proxy.

5.3.2 SOCKS compatible proxy

SOCKS is a standardised proxy protocol for TCP and UDP connections. Our SOCKS-compatible proxy implements the latest version of the protocol 5 ([78]) and the latest draft of its next iteration 6 ([87]), currently under development. A major latency bottleneck of the latest version is the number of required round-trip times (RTTs) during connection setup (*handshake*). In total, this may be up to 5 RTTs. The next iteration reduces the number of required RTTs, introduces minor tweaks to the protocol, and specifies how SOCKS implementations can, optionally, utilise Multipath TCP ([41]) and TCP Fast Open ([24]).

Our SOCKS-compatible proxy implementation, illustrated in Figure 5.5, relies on a redirector component running on all nodes that will be used by the EPIF in the various domains (Section 8.8). Using `iptables`, a packet filtering utility for the Linux kernel[1], the redirector component intercepts all outgoing network connections and redirects them to the proxy. As specified by the SOCKS protocol, the redirector will also communicate the intended destination of connections with the proxy. Based on the source, intended destination, and the area logic model (Section 5.2.1), the proxy can correctly process and forward the connections.

[1] `https://www.netfilter.org`

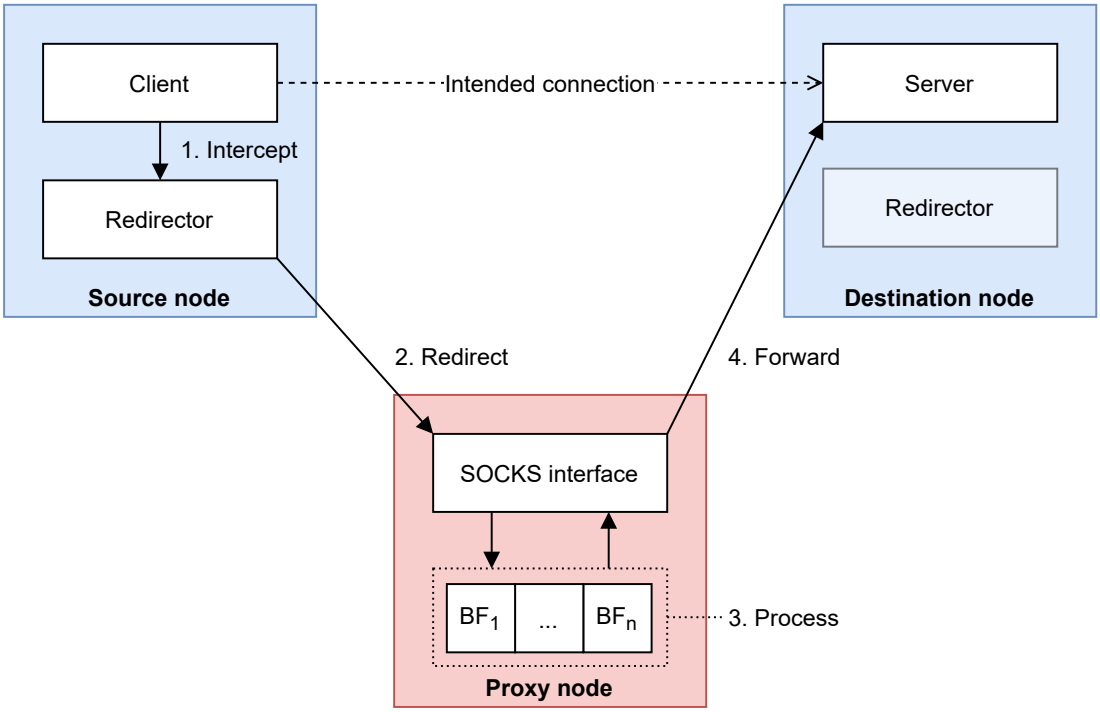


Figure 5.5: The SOCKS-compatible proxy receives redirected connections that the redirector on the client node intercepts.

For the proxy, processing connections include applying the appropriate BFs (Sec. 5.2). The SOCKS protocol can also function as a template for the described BF chain mechanism (Sec. 5.2.2). Specifically, SOCKS proxies are chainable and the intended destination of connections can easily be preserved. BFs can share metadata during SOCKS connection setup(s), i.e., chain establishment, to enable inter-BF coordination. Furthermore, BFs with a uniform interface make dynamic and arbitrary chain compositions straightforward. The accumulative latency bottleneck of SOCKS is reduced using the latest iteration of the protocol and becomes negligible when the entire BF chain runs on the same proxy node.

5.4 Communication with Proxy

To understand the effect of simulating varying network topology distances, we take a closer look at the sequence of communication per each proxy implementation approach. Fig. 5.6 illustrates typical sequence communication steps between a client C and a server S in a no-proxy scenario. Throughout this section, we assume that this is an HTTP request over TCP: SYN/ACK+SYN packets are from TCP and the Request/ Response represents HTTP messages. We formalise Eq. 5.6 to estimate the communication setup time in a no-proxy scenario, such

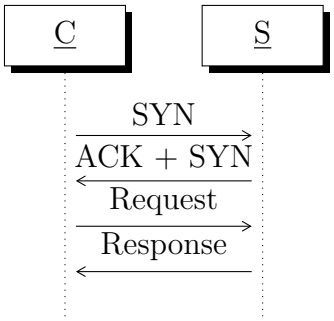


Figure 5.6: A sequence diagram showing the communication steps in a no-proxy scenario.

that:

$$T_{No-proxy} \approx 2RTT_{CS} + t_{server}, \tag{5.6}$$

where RTT_{CS} is the time of a single round-trip time from C to S, and t_{server} is the server request processing time. We use the same server container in all scenarios so t_{server} does not change throughout this paper. $T_{No-proxy}$ is dependent on the CS distance, and it is used as a base reference to calculate Δt in other scenarios involving a proxy.

Fig. 5.7 focuses on the NGINX communication sequence between client C, proxy P, and server S. In this diagram we have node P running between C and S, which introduces extra SYN and ACK traffic. Subsequently, we estimate the

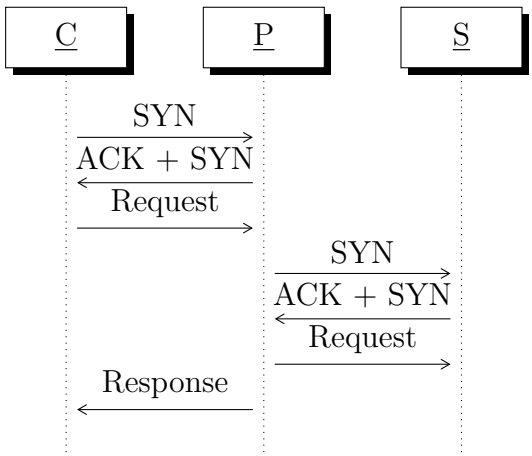


Figure 5.7: A sequence diagram showing the communication steps in a scenario including NGINX-based proxy.

setup time of communication passing through an NGINX-based proxy with:

$$T_{NGINX} \approx 2RTT_{CP} + 2RTT_{PS} + t_{server} + t_{proxy}, \tag{5.7}$$

where RTT_{CP} and RTT_{PS} are the round-trip time between C-P, and P-S, respectively, and t_{proxy} is the processing time of an NGINX proxy.

Moreover, we illustrate a similar communication sequence for SOCKS5-based proxies in Fig. 5.8, where we introduce the redirector element. The redirector always runs on the same host (as shown in Sec. 5.3.2), meaning that the round-trip time from C to redirector is negligible.

There are extra steps of authentication in this proxy implementation, which implies higher overhead, and an overall higher setup time due to additional round-trips.

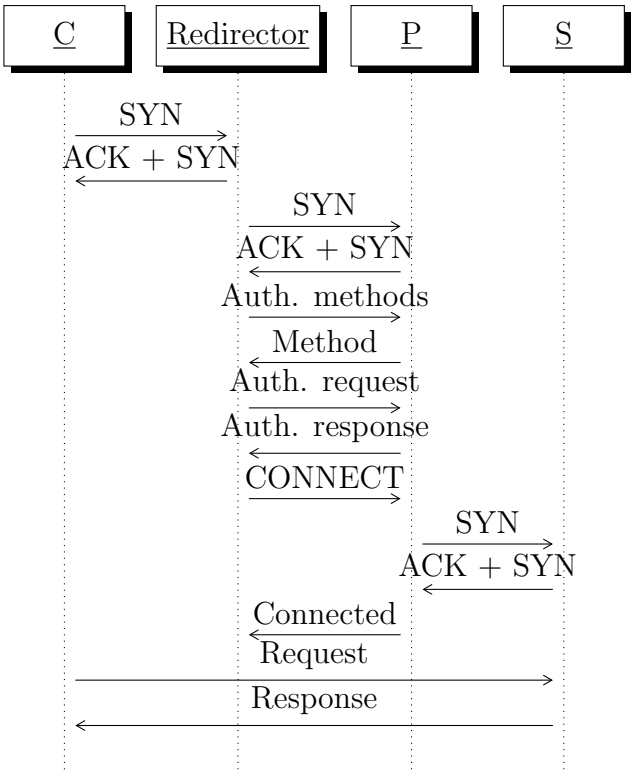


Figure 5.8: A sequence diagram showing the communication steps in a scenario including SOCKS5-based proxy.

The notation of the setup time for a request passing through a SOCK5-based proxy is T_{SOCKS5}, and we estimate it with:

$$T_{SOCKS5} \approx 5RTT_{CP} + 2RTT_{PS} + t_{redirector} + t_{proxy} + t_{server}, \tag{5.8}$$

such that, RTT_{CP} is the round-trip time from C to P (passing through the redirector), and RTT_{PS} from P to S. t_{proxy} is the SOCKS5 proxy processing time, and $t_{redirector}$ is the redirector processing time.

Similarly, Fig. 5.9 shows the communication steps with SOCKS6-based proxy. This proxy implementation optimises the required RTTs, which is apparent compared to the SOCKS5 proxy. The notation of the setup time for a request passing

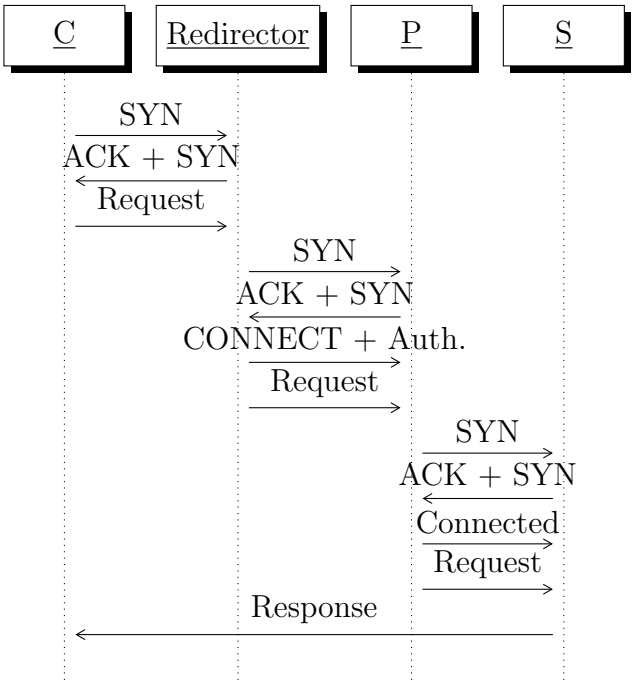


Figure 5.9: A sequence diagram showing the communication steps in a scenario including SOCKS6-based proxy.

through SOCKS6 proxy is T_{SOCKS6}, and we approximate it with:

$$T_{SOCKS6} \approx 3RTT_{CP} + 2RTT_{PS} + t_{redirector} + t_{proxy} + t_{server}, \tag{5.9}$$

where t_{proxy} is the time for the SOCKS6 proxy to process an upcoming forwarding request.

Implementing different proxies implies a different overhead compared to a no-proxy scenario. The evaluation of the two implementations will be detailed in the next section.

5.5 Evaluation

To determine which implementation should be adopted, and under which conditions, we benchmark the two approaches (Sec. 5.3) to evaluate their performance in terms of time overhead and the rate of processed transactions. In our experiments, we fully containerise and automate the benchmark setup in Docker containers for reproducible results. Our benchmark implementation is publicly available online [2].

[2] https://github.com/epi-project/proxy-bench

5.5.1 Experiment topologies

We run three different application containers: client, web server and proxy. We generate request traffic from the client to the server passing through the proxy and use the no-proxy scenario as the baseline for this benchmark, using two network tools: *httping* (3) and *wrk* (5). We are interested in the additional time that packets take to go through the proxy and give it the Δt notation.

In an attempt to have a controlled environment with reproducible results, we dockerise everything and run it all the application components on a single VM machine. We configure different network configuration combinations; we accomplish this by varying the distance/latency between containers using `tc`. By doing this, we mimic a real network with changing distances between nodes.

We run several network typologies scenarios on a basic Debian version 10 VMs, with 2 cores, 2 GB of RAM, and 20 GB of storage. In one topology, the proxy is placed between the client and the server; we call this the *proxy-in-between* topology. In a second topology, the proxy is placed at a location that is equidistant from the client and server; we call this the *triangular* topology. The two types of setup are illustrated in Fig. 5.10. Namely, CP represents the client-proxy distance, PS the proxy-server distance, and CS the client-server distance (as defined in Sec. 5.4).

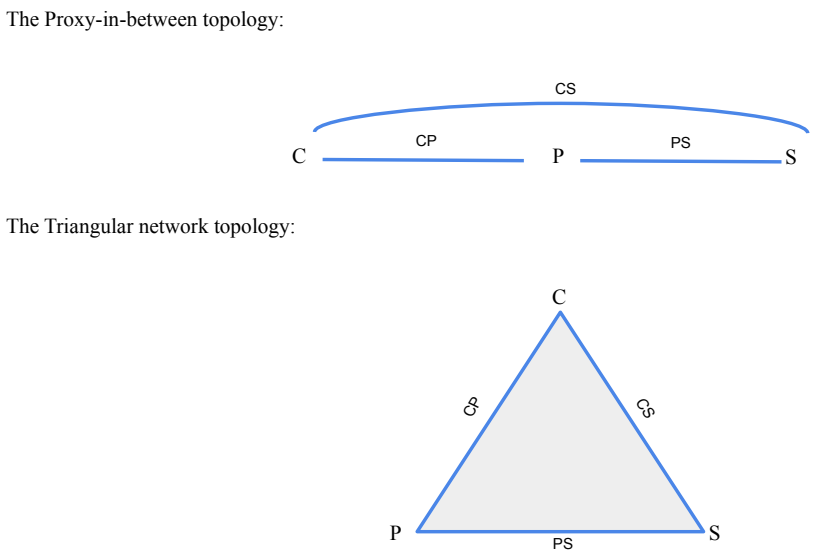


Figure 5.10: The different network distance/latency typologies with C representing the client node, P the proxy node, and S the server node.

The *tc* tool, a network control utility for the Linux kernel, is used to simulate different distances between the nodes to evaluate the effect of varying delays on

the time overhead. Table 5.1 shows the six different latency configurations we have adopted.

Topology	Name	CP (ms)	PS (ms)	CS (ms)
Proxy-in-between	DOCKER 1	5	5	10
	DOCKER 2	5	10	15
	DOCKER 3	10	5	15
Triangular	DOCKER 4	1	1	1
	DOCKER 5	5	5	5
	DOCKER 6	10	10	10

Table 5.1: The six network configurations used in our experiments and the respective latencies; three topologies (1-3) are related to *proxy-in-between* setup and three topologies (4-6) are related to the *triangular* setup

5.5.2 Proxy-in-between topology experiments

In the first experiment, we place the proxy in between and on the direct path of C and S, and simulate distance differences between the nodes according to the values in Table 5.1: DOCKER1, DOCKER2, DOCKER3. We instantiate the client, proxy, and server containers on a VM and we generate HTTP traffic via the *httping* tool. Then, we measure the average round-trip time of 120 consecutive requests.

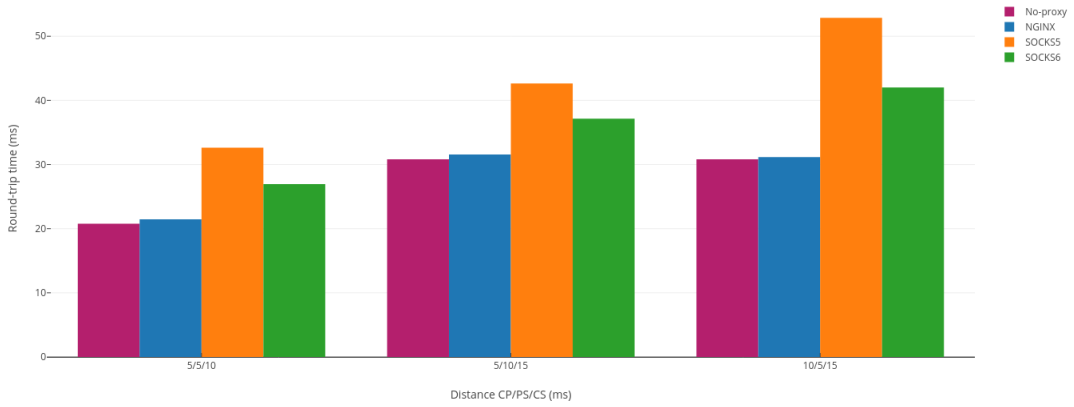
The results of the measured round-trip time (ms) and the associated overhead Δt are plotted in Fig. 5.11: 5.11a and 5.11b, respectively. As the total distance between client and server increases, so does the total round trip time (Fig. 5.11a).

In Fig. 5.11b the observed Δt of NGINX is $< 1ms$ when it is deployed in between the client and server. If the SOCKS6 proxy is halfway of the distance, then the Δt is $\simeq 6ms$. As for SOCKS5 proxy, Δt is $\simeq 12ms$ and it increases rapidly with the CP distance.

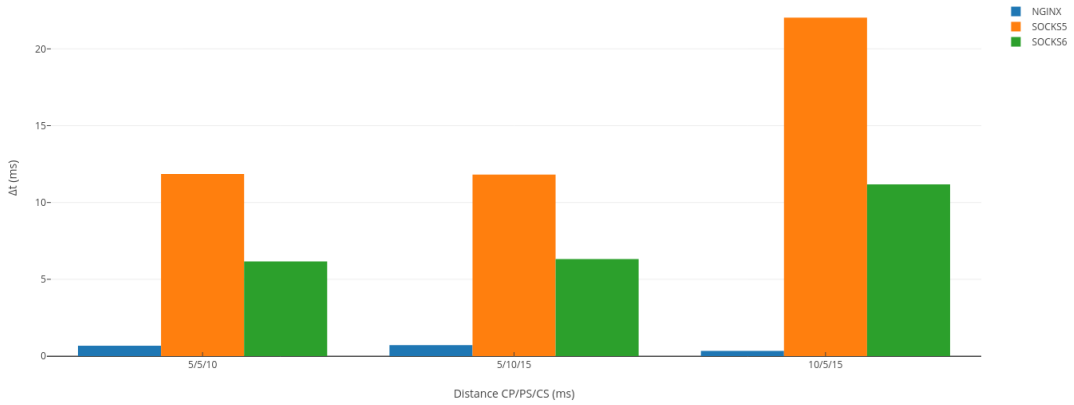
5.5.3 Triangular topology experiments

In this experiment, we simulate equidistant triangular topologies with CP/PS/CS values shown in Table 5.1: DOCKER 4, DOCKER5, and DOCKER6.

The plots in Fig. 5.12 show the resulting round-trip time and Δt with 5.12a and 5.12b, respectively. We observe that the resulting overhead of the proxies implementations increases with distance. The overhead values are higher than the values recorded with the proxy-in-between scenarios, ranging from 2 ms to 21



(a) The average of the client to server 120 consecutive requests round-trip time (ms) with changing configured distances between nodes.



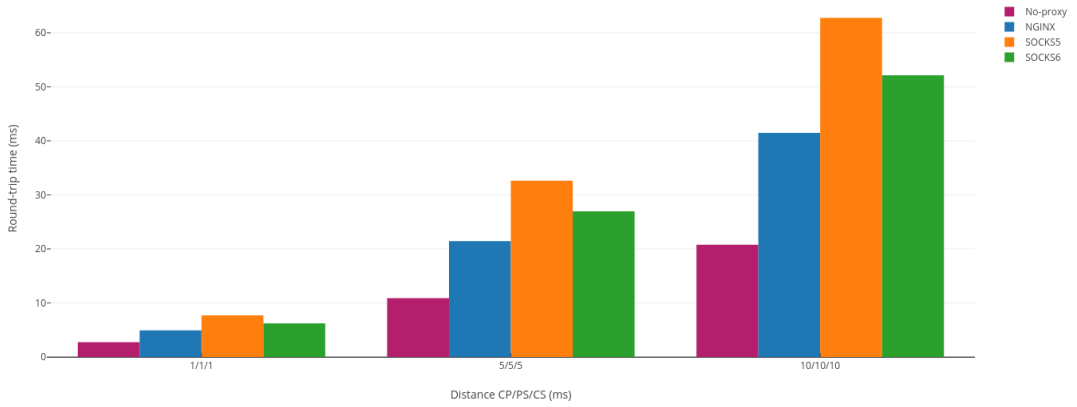
(b) The overhead of Δt (ms) of different proxy implementations compared to no proxy with changing configured distances.

Figure 5.11: The variation of round-trip time (ms) and overhead Δt (ms) of proxied HTTPING requests with proxy-in-between network topologies.

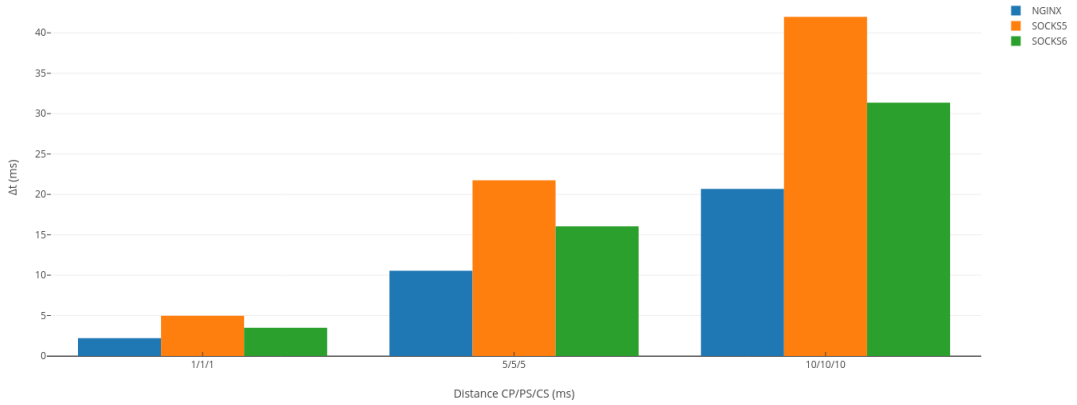
ms for NGINX, from 5 ms to 42 ms for SOCKS5, and 4 ms to 32 ms for SOCKS6 (as shown in 5.12b).

5.5.4 Rate of processed transactions

To test the rate of processed transactions of each proxy implementation we set up a third experiment. In this case, we utilise 3 different VMs each running an application container (client—proxy—server). The reason that we don't containerise



(a) The average of the client to server 120 consecutive requests round-trip time (ms) generated via HTTPING with changing configured distances between nodes.



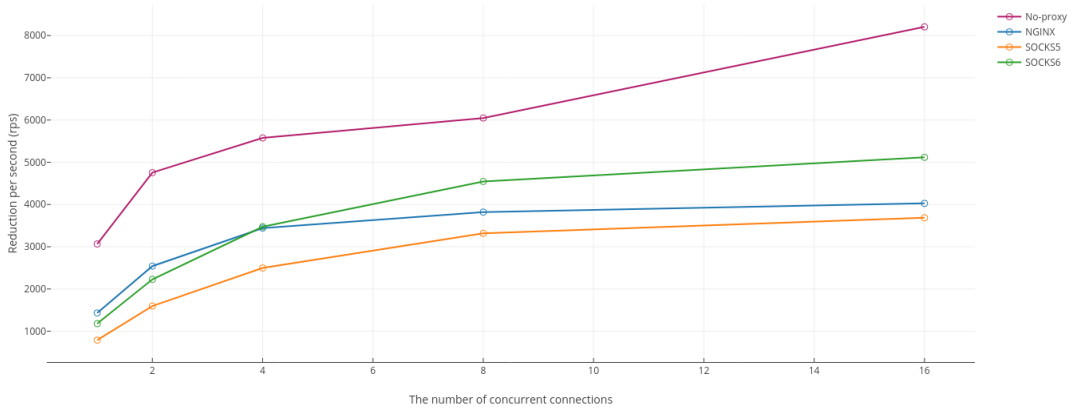
(b) The overhead of Δt (ms) of different proxy implementations compared to no proxy with changing configured distances.

Figure 5.12: The round-trip time (ms) and overhead Δt (ms) of proxied HTTPING requests with triangular network topologies.

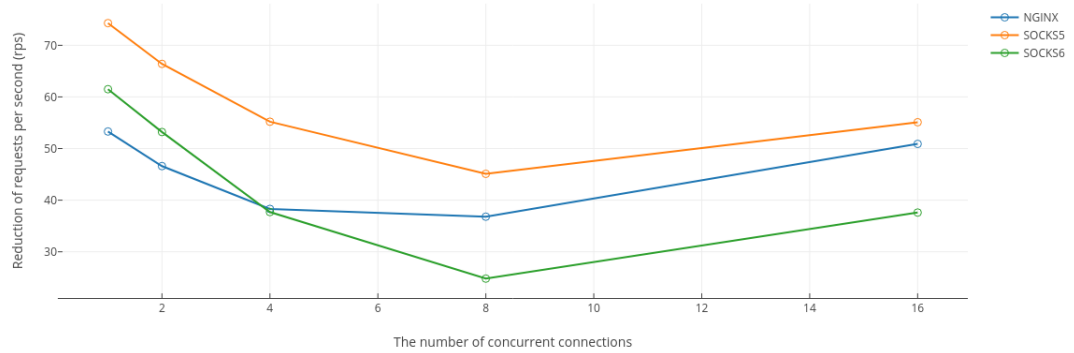
this setup is that with the *wrk* tool, applications will compete for resources and we'll end up with lower rate results.

Fig. 5.13 shows the rate of processed transactions of each proxy implementation measured with *wrk* as the number of concurrent HTTP connections increase (Fig. 5.13a) and the reduction of this rate compared to the no-proxy output (Fig. 5.13b). The network setup behind this plot simulates no varying distances.

Similar to the no-proxy plot, the rate of processed requests of all three implementations increases with the increasing number of connections, but it flattens



(a) The rate of processed transactions resulting via *wrk* of different proxy implementations with increasing concurrent connections.



(b) The reduction of processed requests per second of different proxy implementations compared to no-proxy.

Figure 5.13: The number of HTTP requests that are processed per second (rps) and the associated reduction with the increasing number of concurrent requests.

when it hits 8 concurrent connections. That might be explained due to the proxy reaching a bottleneck of resource consumption.

The bottleneck is further reflected in the second plot in Fig. 5.13b, the reduction of the rate compared to the no-proxy rate decreases initially to show that the processing rate of the three proxy implementations is growing at a higher rate. When we reach 8 concurrent connections, the processing rate flattens out while the no-proxy rate is still increasing. Subsequently, the reduction rate of the proxies starts to increase slightly afterwards.

We can concur from this plot that next to no-proxy, the SOCKS6 approach

results in the highest processing rate with increasing connections. Moreover, the SOCKS5 approach has the lowest processing rate and doesn't scale as well compared to SOCKS6 and NGINX.

5.5.5 Discussion

In Fig. 5.11b The proxies' overhead increases with the increasing distance between the proxy and the client, while it does not noticeably differ with increasing distance between the proxy and server. Such a behaviour is logical and expected given the contributions to the total overhead of the various components as explained in Sec. 5.4.

While in Fig. 5.12b we notice that the NGINX proxy performs better in terms of Δt in both setups. However, the SOCKS-based proxies imply more overhead, which is expected since they need more authentication steps during connection setup (handshakes), while the NGINX proxy simply forwards requests.

The overhead resulting from these experiments varies according to $\Delta t = T_{NGINX|SOCKS5|SOCKS6} - T_{No-proxy}$. This explains the smaller overhead in the proxy-in-between topologies that implies higher RTT_{CS} time compared to triangular topologies.

Experiments in Sec. 5.5.2 and 5.5.3 show that the placement of the proxy node has a noticeable effect in terms of overhead. Moreover, it is proven to be a way of minimising the inevitable latency of introducing proxies.

5.6 Comparison

There are other evaluation performance parameters of each proxy that we want to consider. Here we look at the port scalability, optimisation, portability and reconfiguration, dynamicity, complexity and security.

Table 5.2 shows the comparison between different proxy implementations according to the previously defined parameters.

5.6.1 Port scalability

Due to their different mechanisms, different numbers of ports need to be opened on a proxy server. As an example, we need one port for each proxy for $SOCKS(5|6)$, while one port per node pair of nodes for NGINX. This reflects on the hardware scalability of the proxy with higher requesting nodes.

5.6.2 Optimisation

NGINX is optimised for HTTP (more optimised in general as well), while SOCKS works on top of the TCP/UDP transport layer. This is reflected in the results of

Parameters	NGINX	SOCKS5	SOCKS6
Δt	✓		
Processing rate			✓
Port scalability		✓	✓
Reconfiguration		✓	✓
Dynamicity		✓	✓
Security		✓	✓

Table 5.2: The comparison between different proxy implementations according to six performance parameters; where the ✓represents an advantage over other proxies.

the experiments illustrated in Fig. 5.12 and 5.11.

5.6.3 Reconfiguration

NGINX is configured through configuration files, and subsequently, reconfiguration of the service might require a restart (downtime might be avoided through canary (re)deployment). On the other hand, SOCKS configuration is dynamic/ programmable, and it is updateable without restart/ downtime, *e.g.* database entry update.

5.6.4 Dynamicity

NGINX paths are static (based on the incoming port used), thus BF chains are static as well. Separate BF chains (x containers) need to be deployed per node pair, thus scaling is also done per node pair BF chain. With SOCKS the destination is communicated as part of the connection setup, thus BF containers can remain stateless (assuming a BF implements a SOCKS interface). Meaning that one BF container can be part of multiple BF chains *e.g.* through dynamic function composition $g(f(x))$ based on SOCKS chaining. Scaling can be done per BF across different node pair paths. (i.e. scale per number of connections to the proxy, instead of the number of connections per node pair.)

5.6.5 Security

From a security perspective, SOCKS authentication is part of the protocol. Whitelist, mutual TLS, and/or shared secret are supported. That is not part of SOCKS but

is possible and handled by the redirector component. On the other hand, with NGINX whitelist and mutual TLS are supported, but it is not transparent to the client app.

5.7 Related work

Different redirection tools have been implemented in the past to fulfil a number of different purposes. In (40) they propose a dynamic hybrid honeypot system based on a transparent traffic redirection mechanism to address the identical fingerprint problem. They implement the Honeybrid gateway to capture data and control traffic. The Honeybrid gateway includes a Decision Engine and a Redirection Engine, which are in charge of orchestrating the filtering and the redirection between frontends and backends. The Decision Engine is used to select interesting traffic, and the Redirection Engine is used to transparently redirect the traffic. They employ a traditional TCP proxy that applies the TCP relay mechanism. In our case, redirection relies on the logic area generator, which has deep knowledge and integration with the policy engine; the Decision Engine in (40) would not be easy to adopt. Likewise, the Redirection Engine also provides an orchestration function, which in the EPIF are already integral components of the application and infrastructure orchestrators.

For similar security reasons to ours, (4) employs a Subscriber Traffic Redirection software to redirect HTTPS requests via a redirection server. Similar to our proposed implementation, the server redirects traffic to a previously configured web server with a Secure Sockets Layer certificate (SSL). This tool is evaluated according to security parameters, such as security against DDoS attacks. In our case though, we need a proxy that is capable of redirecting all kinds of application traffic, not only HTTPS.

Finally, (72) evaluates the high availability of two proxy implementations: reverse proxy and SDN-based proxy, based on OpenFlow. The conclusion is that the SDN proxy proves to be robust against link failure due to its ability to monitor network interfaces. The idea of using an SDN-based proxy is certainly relevant to our effort to exploit programmable infrastructures. Still, OpenFlow is no longer the main SDN technology, and it must be noted that most of the current EPI infrastructures would not have OpenFlow devices running in them. We foresee to evaluate data plane programmable solutions such as P4.

5.8 Conclusion & Future work

Interception and redirection of traffic is a core feature of the EPIF. In this chapter, we evaluated and benchmarked the setup time of two different approaches; an NGINX-based reverse proxy method and a SOCKS-based method.

We determined that the overhead of the EPIF proxies differs in the various implementations, and it relates directly to the positioning of the proxy within a network topology. Other than overhead, we also considered different performance parameters in evaluating the proxies implementations. We can conclude that the SOCKS6-based proxy offers the highest processing rate (indicative in throughput with the proxy intercepting and forwarding traffic), and it supports all traffic type redirection. In addition to that, SOCKS proxies have advantages in terms of reconfiguration, dynamicity, security, and scales better in employing open ports. On the other hand, the NGINX-based approach processes forwarding requests with lower overhead.

Subsequently, to make a design decision on which proxy implementation should be used, we need to consider the type of application being served. Namely, the choice depends on the application requirements and the specific relevance of performance parameters. For example, a data streaming application would benefit from a SOCKS6-based proxy because of the higher processing rate. Subsequently, this will lead to minimal overhead

In the next chapter, we will be implementing more EPIF functionalities like BF chaining and offering uniform interfaces of bridging functions. This would enable programmability. Our focus will therefore be put on the extra plug-ins needed in the redirection tools needed for BF's chaining. We will focus our work on computing efficiency, and build the CPU profiles of different BFCs to recommend optimal chaining and placement.

Chapter 6

Bridging Function Chains: Resource Profiles

Different use cases require different network configurations, different performance requirements, and computing resources. Addressing the broader challenges of network performance and resource limitations within healthcare data sharing, this chapter systematically profiles the resource utilization of SFCs to enable data movement across different healthcare use cases. We provide example configurations that map to a couple of use cases (e-Health record query and heath data streaming), and then we monitor and collect CPU utilization of the different CNF (Containerised Network Function) compositions. In the considered policies, we can: discard flow, protect (encrypt) and transmit, or allow with no protection. To enforce each policy, we deploy a firewall function (relatively heavy-weight function), an encryption function, and a decryption function (light-weight stream cypher). We analyse the behaviour of the SFC microservices with various setups, and we aim to further use this analysis to build the placement heuristic according to available and trusted cluster resources. Subsequently, these profiles will be utilized in the next chapter to recommend heuristic-based placement based on collected profiling data of resource usage and limits for high availability, optimal performance, and minimal resource waste. This chapter helps in answering the question:

RQ2: "What are the performance tradeoffs when deploying different packets' redirection methods and bridging functions to enforce network policy-compliant routes under different workloads?"

This chapter is based on:

- **J. A. Kassem**, A. Belloum, T. Müller and P. Grosso, "Utilisation Profiles of Bridging Function Chain for Healthcare Use Cases," 2022 IEEE 18th International Conference on e-Science (e-Science), Salt Lake City, UT, USA, 2022, pp. 475-480, doi: 10.1109/e-Science55777.2022.00085.

6.1 Introduction

In line with Next-generation networks, network functions are getting increasingly virtualized and highly programmable. NFV decouples functions; such as packet encryption and firewalling; from specialized hardware to make services deployable on general-use virtual servers (115). This allows on-demand management, dynamic shipment and placement of services, and can potentially provide reliable network performance.

NFV offers the framework to configure, chain, manage, and orchestrate functions according to agreed network policies. Furthermore, containerizing these functions can accomplish fast deployment, high reusability, and low setup overhead (28). The controlled service scheduling and placement, namely containerized NFV-based (CNFV) services, is a functionality that aligns with what we are trying to accomplish in supporting different health use cases. We simulate two of the use cases by simulating different workloads and profiling resources:

- **Use case 1**: Hosting a centralized health data registry and maintaining health data entries of different healthcare institutions, and the user can query an entry in the dataset.
- **Use case 2**: Streaming data via IoT devices and wearable to monitor the users' health and provide timely intervention.

The BFC orchestrator employs CNFV to implement network service chains, and we configure the setup to adapt to different healthcare use cases' policies and requirements. The policies describe the communication constraints, traffic filtering rules, trust, and the required Network Functions that need to be instantiated and deployed. The framework can also be used with use cases other than healthcare by using the same framework's handles to generally translate and map policies to the VNF chain. To implement the security services, we need middleboxes hosting the NF's; known as proxies. The proxied services can vary from lightweight to heavyweight functions, depending on the type of function and the implementation's optimization. Moreover, the resource consumption behaviour also depends on the expected workload whilst running a specific use case. Hence, we need to build a profile of the resources required under labelled use cases and

the associated network policy. Furthermore, we have to allocate sufficient computing resources to ensure smooth deployment and high availability throughout the execution of the application. The challenge that still remains is the optimal and adaptive placement and configuration of these services to available resources across trusted clusters of proxy nodes.

6.2 Related Work

With similar research efforts, (16) (27) propose policy enforcement systems using SDN tools (OpenFlow), to "steer traffic" through a policy-defined service chain. To reduce the resources needed by the middleboxes, they define ways to optimise the switch table sizes and flow control rules. On the other hand, employing OpenFlow switches can potentially introduce new challenges, such as dedicating specialised hardware and having to configure and add SDN components to the infrastructure.

Subsequently, adding virtualisation capabilities is needed, and resource allocation must be effectively orchestrated and managed in order to avoid over/under-provisioning, and to maintain end-to-end latency that is equivalent to those seen in conventional networks. It is crucial to ensure the network service's resilience, knowing that the implementation can get increasingly complex (like firewalling with intensive rules, capturing application-level semantics, and potentially deep packet inspections). According to (43), overloading middle boxes (e.g. proxies) is a typical reason for service failure.

A plethora of work is done on NFV chaining and placement, but different strategies are employed. As an example, the MIDAS (10) framework places the services chain according to the cross-border on-path strategy, where each service is location-specific (*e.g.* a web proxy needs to be placed near the end-client). On the other hand, when the path is loosely controlled, then the placement really depends on the hardware resources available, and that is implemented using varying algorithms (*e.g.* the least busy host placement algorithm (25)). Others consider QoS-driven optimisation placement of the NFV. The used methods vary from using heuristic algorithms (55), Linear programming, and ML-based tools (*e.g.* Reinforcement learning).

In an effort to optimally deploy and maintain the CNFV chains, we add with this work-in-progress to existing literature the defined deployment stages after a network policy is formalised. The placement is decided based on available trusted proxy clusters matched with the profiled data, knowing the labelled use case running. We showcase the exploratory profiling results for designing the heuristic algorithm.

6.3 The Function Chains Deployment

In previous work ([68]), we proposed the EPI framework (EPIF), a dynamic framework that automates the programming of the underlying networks to secure health data-sharing. Data-sharing is secured by orchestrating and managing CNFV; called *Bridging Functions* (BF); to add security value to each communicating node, irrespective of each node's computing capabilities. The BFs can be instantiated and chained to form a *Bridging Function Chain* (BFC) and enforce increasingly complex policies.

The *adaptation* of the underlying networks is done after querying network policies and translating them into setup actions, for example, party A can talk to party B if A is able to encrypt and decrypt via a stream cypher. The automated setup of the infrastructure is also required to achieve reachability of the endpoint nodes, optimal security across collaborating domains, reasonable network performance, bridging services availability, hardware selection and scalability, and ultimately, abiding by policy requirements.

EPIF has different components that are crucial to automating the data-sharing processes between participating parties. The main components of the framework are the application orchestrators, the infrastructure orchestrator, the policy management system, and the logic area generator. The two orchestrators are decoupled (conceptually) to orchestrate and manage different types of functions; the workflow functions, and the network functions, respectively. The policy management system is used as a reasoner to query policies and maintain agreement. The logic area generator ([64]) is in place to assign security areas to end nodes, and subsequently deduce security requirements and policies. The proxy node acts as a framework actor, where the BFC pool is hosted and instantiated when ordered by the orchestrator.

After resolving the BFC rule per source-destination flow, the EPIF instantiates and deploys the services chain to secure connection according to the set policy. The challenge of optimally provisioning these services remains, and so we define with this work three stages for optimal placement and performance of the chain. Fig. 6.1 illustrates these steps, starting with introducing a new healthcare use case with the expected associated BFC workload. Then we start the profiling stages where we test the setup, collect resource metrics via the metric server, calculate statistical information, and store the labelled data for the next stage. The heuristic placement queries the available *trusted* proxy clusters via the cluster manager and matches the required resources with the currently available resources. Trust is modelled according to the node's competence, and the contractual agreement of the institution it runs within. The orchestrator controls the placement of the service chain accordingly to maximise trust and resource utilisation. This would achieve optimal placement of BFC (or semi-optimal, considering inaccurate profiling in case of a small number of test-runs with high standard deviations), and serves as a booster to the Q-learning agent in the next stage. Next,

the run-time maintenance utilises a Reinforcement Learning agent and is asked to monitor QoS metrics (latency, throughput, etc.) and reactively scale in/out. The placement of the replicated scaled-out services can be done across the multiple trusted proxy clusters (not just the chosen placement cluster).

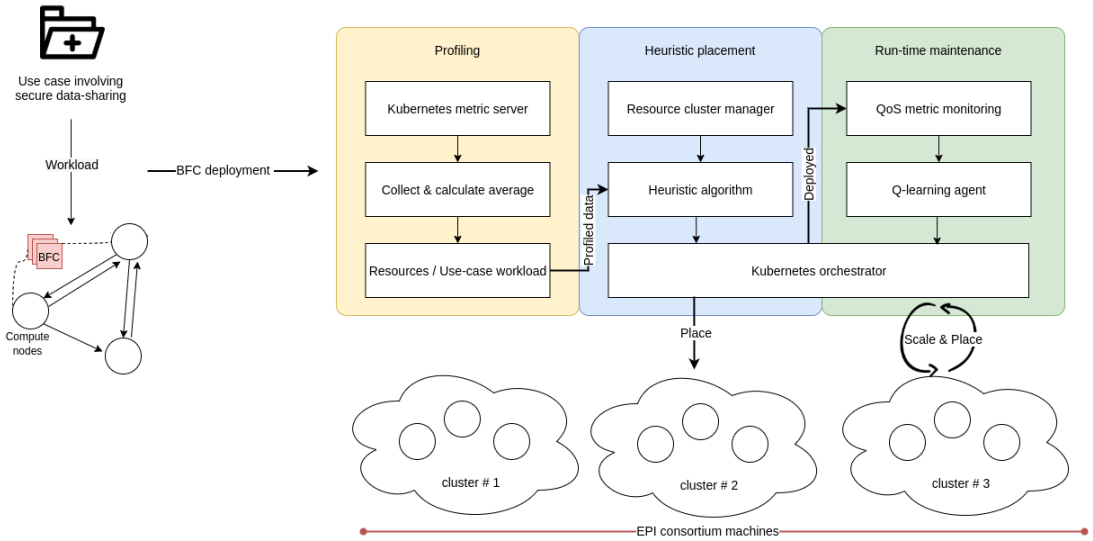


Figure 6.1: The BFC deployment stages after introducing a new health use case: Profiling, Heuristic placement, run-time maintenance.

A simple BFC deployment is illustrated in Fig. 6.2, where we have two types of nodes:

- The master node: acting as the infrastructure orchestrator where we can configure, deploy, and orchestrate microservices
- The worker node: acting as the proxy node hosting and running the BFC. The worker nodes are spread over two different clusters A and B both running on UvA machines in this case.

Fig. 6.2 also shows the example BFC implementations and the proxy service. The proxy service has an important functionality of traffic intercepting and redirecting to route through the BFC. With that we eliminate the need to manually plan, compose, and configure routes across the service chain (example: $Client \rightarrow NF_1 \rightarrow NF_2 \rightarrow NF_3 \rightarrow Server$), by introducing the SOCKS(5|6) ([79]) ([87]) proxy that intercepts and reroutes packets seamlessly to the end-points nodes.

We focus in this paper on the first stage, and we populate the profile data storage with resources consumed running use cases 1 and 2.

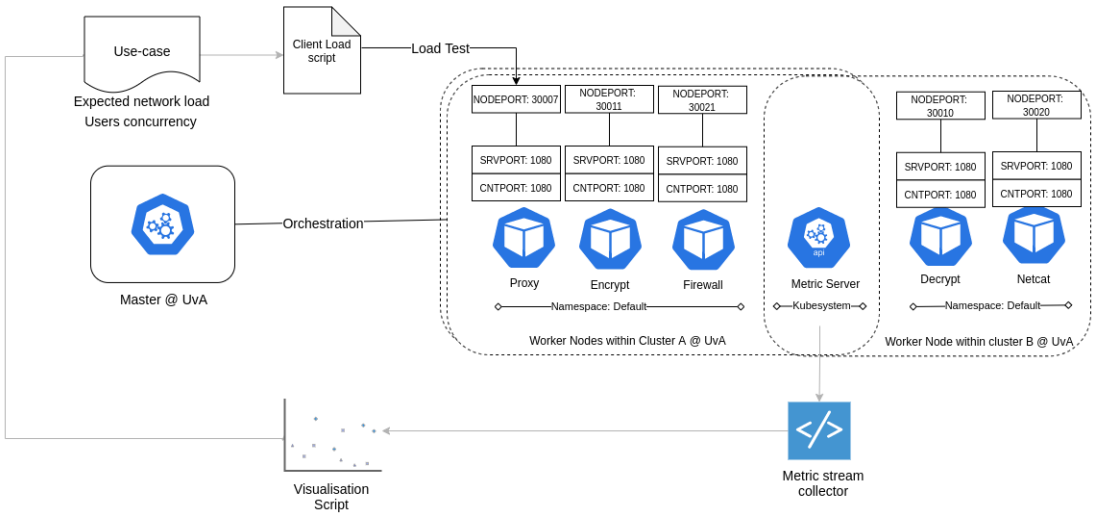


Figure 6.2: The setup of the EPI function chains at UvA with one master node as the orchestrator and the worker nodes across cluster A & B hosting the topology of BFC.

6.4 Experiments

We design the following experiment to evaluate the CPU usage of different BF chains. We do not consider memory because our cases do not pose a strain on the resources. That is due to the type of functions implemented. We deploy this framework (shown in Fig. 6.2) using four identical Ubuntu 18.04 VMs with 2 cores, 2 GB RAM, and 20 GB HD. The VMs are hosted on a UvA computing server cluster, and the VMs are connected via the same network. In our effort to run this in a controlled environment, we containerise all services and orchestrate with Kubernetes. All the needed services run on the worker nodes and that is made available via dedicating an external port that can be pinged and reached from outside the node.

The SOCKS6 proxy we implement is a standard proxying protocol for TCP and UDP connections. SOCKS6 is an optimised iteration of SOCKS5, and it introduces minor tweaks to the *Authentication handshake*. The stateful Python-based firewall function cross-checks the packets across a couple of rules to either accept or reject traffic, then redirects it back to the proxy to route to the next stop (next BF or destination). We implemented the encryption and decryption functions with ChaCha20 stream cipher ([85]). It is a lightweight function that improves on the Salsa algorithm ([58]) by increasing per-round diffusion without decreasing the performance.

The metric server is a cluster-wide data aggregator of resource usage and will scrape the relevant resource metrics of each pod. We use the metric server to gather 100 samples of the CPU usage/pod, and then calculate statistical values

(average and standard deviation error). All relevant code is made available on GitHub[1] for reproducibility purposes.

We assign a fixed CPU request and limit to all containers of 500milliCores and 600milliCores, respectively. We run four different configurations. The first use case is the one with a lower sending rate of 100kB/s, and we rerun this experiment twice with 1 then 10 concurrent clients. Similarly, we also run the same experiment with a use case simulating a heavier workload of 1000kB/s. We purposely generate traffic that will be accepted and forwarded, because we need traffic to propagate across the chain

6.5 Profiling Results

The varying composition, length, and order of the functions within a chain can result in different CPU consumption per pod. That is further apparent in Fig. 6.3, Fig. 6.4, Fig. 6.5, Fig. 6.6.

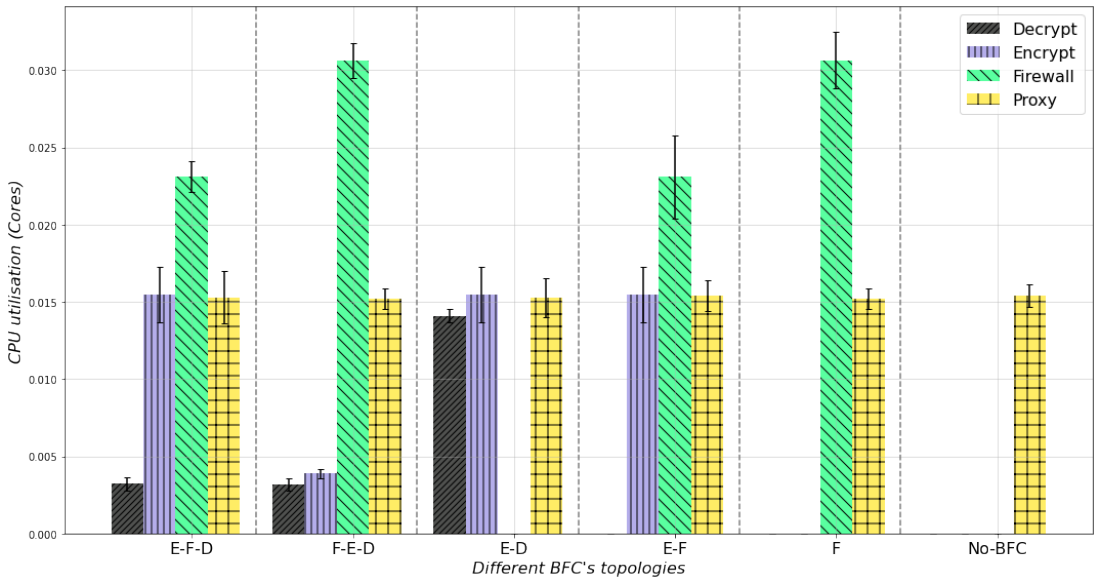


Figure 6.3: Use case 1: CPU utilisation per pod and the throughput at the end server with one client generating 1000 kB/s traffic, and grouped by service BFC topologies composed of encryption (E), decryption (D), and firewall (F) services.

Considering the proxy service, the CPU utilisation does not change across topologies. This is due to the minimal proxy's functionality of redirecting traffic according to the chain, and the fact that most processing is done on the NIC (Network Interface Card).

[1]https://github.com/epi-project/EPI-kube-scaling

While the other services are affected by the use case applied and the number of concurrent users using the system. First, the firewall service has the highest CPU utilisation being the heavier-weight function. Compared to the other topologies, the consumption average is higher within the F-E-D and F chains, and that is related to the first order of the service.

Second, the encryption and decryption services expectedly have approximately equal CPU usage across the F-E-D and E-D topologies. That is due to similar compiling instructions and code implementation. Unlike with the E-F-D topology, where the decryption service has a much lower CPU average. This is caused by the ordering of the firewall in the middle of the chain. This reduction is caused by the different processing rates at the firewall, and this can differ with different implementations (larger number of rules, code optimisation, etc.) Furthermore, the reduction is also affecting the needed CPU power by both services (encryption and decryption) in the F-E-D chain. Similarly, the causing factor is the firewall placement at the beginning of the chain.

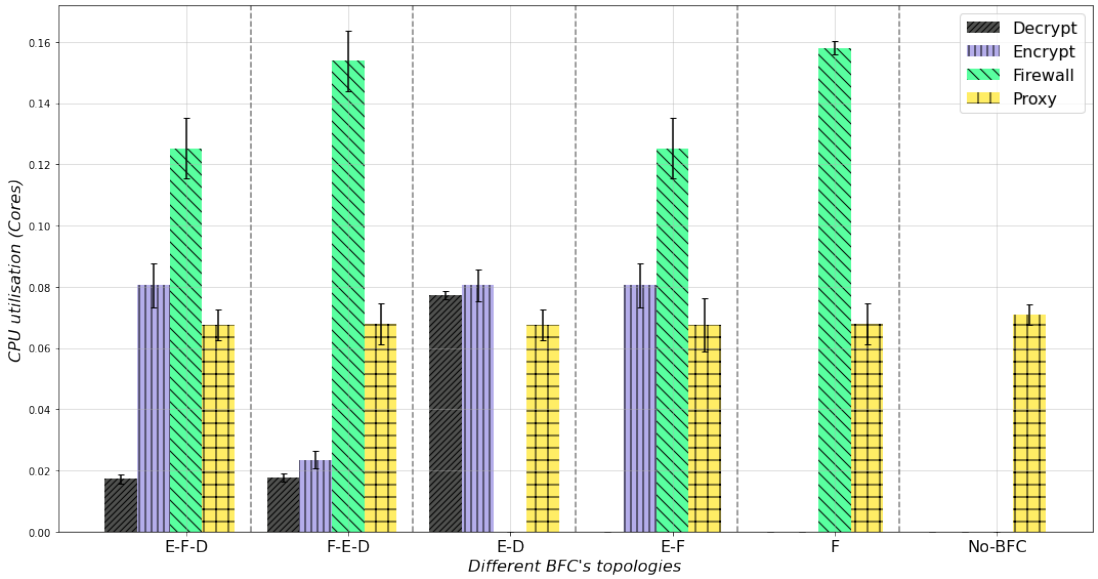


Figure 6.4: Use case 1: CPU utilisation per pod and the throughput at the end server with ten clients generating 1000 kB/s traffic, and grouped by service BFC topologies composed of encryption (E), decryption (D), and firewall (F) services.

Intuitively, the CPU utilisation of the services varies with the different use cases, and increases with more clients, while the behaviour is consistent throughout the experiments. These values are generated automatically after initiating a profiling stage, and this serves as an input to the next stages.

To prove the relation between the BFC topologies' composition and the received packets' rate reduction, we measure packets received per second at the

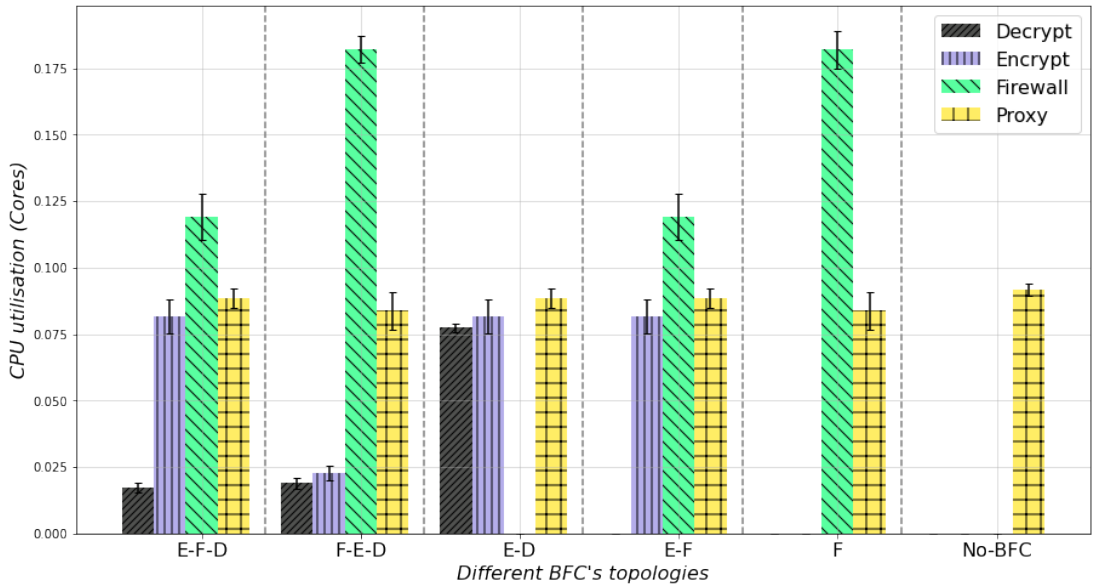


Figure 6.5: Use case 2: CPU utilisation per pod and the throughput at the end server with one client generating 1000 kB/s traffic, and grouped by service BFC topologies composed of encryption (E), decryption (D), and firewall (F) services.

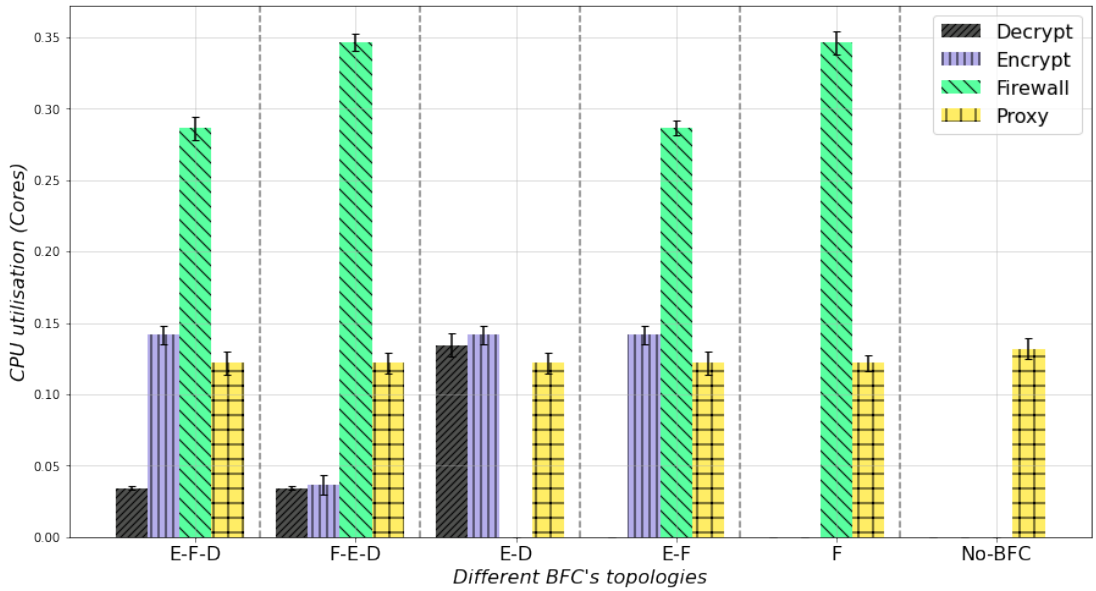


Figure 6.6: Use case 2: CPU utilisation per pod and the throughput at the end server with ten clients generating 1000 kB/s traffic, and grouped by service BFC topologies composed of encryption (E), decryption (D), and firewall (F) services.

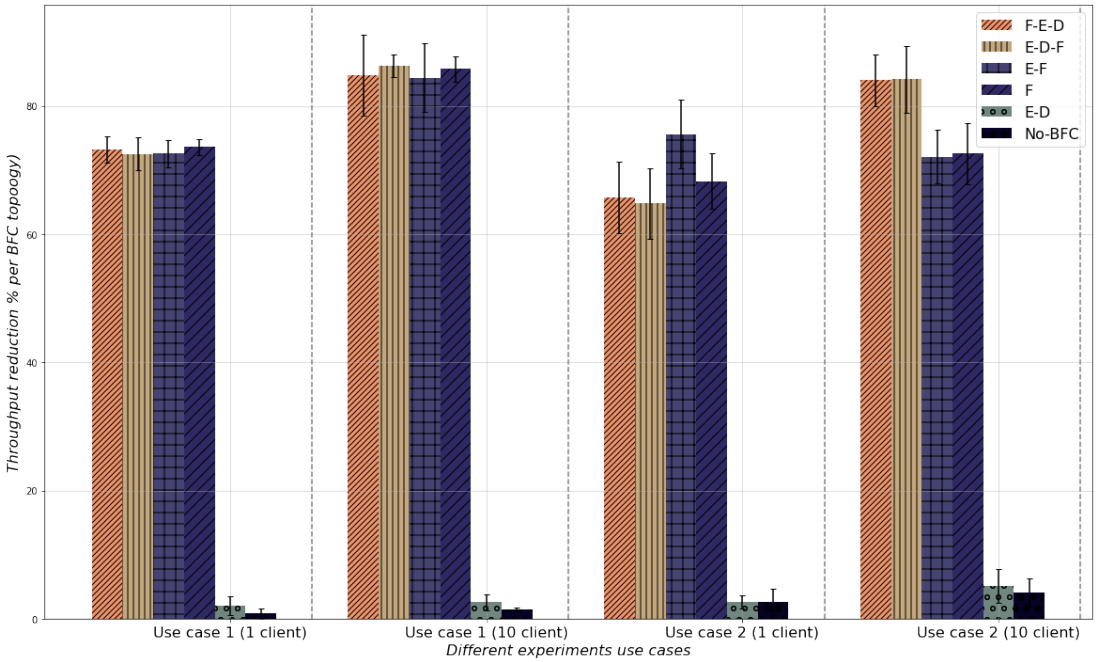


Figure 6.7: The monitored throughput reduction profile for the different use cases with different BFC setups

Socat end-server. Fig. 6.7 shows the throughput reduction percentage/topology while running the four experiments. The topology chains with no firewall have a much higher throughput than the other setups.

This throughput is slightly lower than having no BFC at all. To counter this effect, we employ a QoS-based scaler that monitors the throughput across each function and scales out (or in) by replicating the service (horizontal scale) and placing the pods according to the heuristic.

6.6 Conclusion

In this chapter, we provided insights into the resource utilization trends across SFC microservices. Moreover, we built the handles to monitor services' performance, and that's the groundwork for the other stages of optimal deployment. In our next chapter, we aim to use these data to automatically provision the microservice across trusted clusters and update its configuration. We will also introduce the scalers to the setup. The scaler proposed will be triggered with a Heuristic-Boosted Q-learning agent. We plan to compare the results of traditional provisioning mechanics to the proposed heuristic-boosted Q-learning one. The comparison will be done according to three metrics: the QoS performance of the microservice chains, packet loss, and the deployment success rate.

Chapter 7

Bridging Function Chains Provisioning and Placement

In this chapter, we highlight the efficiency of the infrastructure orchestrator by provisioning adaptive VNFs to enforce security policies associated with different data-sharing scenarios (data-sharing includes algorithms and processed data). We define a Cloud-Native Network orchestrator on top of a multi-node cluster mesh infrastructure for flexible and dynamic container scheduling. The proposed framework considers the intended data-sharing use case, the policies associated, and infrastructure configurations, then it provides SFC and provides routing configurations accordingly with little to no human intervention.

Moreover, what is *optimal* when deploying SFC is dependent on the use case itself, and we tune the provisioning algorithms' hyperparameters to prioritise resource utilisation or latency in an effort to comply with the performance requirements. As a result, we provide an adaptive network orchestration for digital health twin use cases, that is policy-aware, requirements-aware, and resource-aware. In this chapter, we answer the question:

RQ3: "How to automate an adaptive Service Function Chain Provisioning on available network Points of Placement candidates to run a data-sharing request under different use cases' requirements?"

This chapter is based on:

- **Jamila Alsayed Kassem**, Li Zhong, Arie Taal, and Paola Grosso. "Adaptive Services Function Chain Orchestration For Digital Health Twin Use Cases: Heuristic-boosted Q-Learning Approach," NetSoft2023.

7.1 Introduction

The widespread adoption of DTs in healthcare faces substantial challenges, including stringent health data-sharing policies, high-performance network demands, and potential limitations in infrastructure resources. In this chapter, we aim to confront and overcome these challenges through the implementation of adaptive VNFs dedicated to enforcing security policies tailored to various data-sharing scenarios.

At the centre of our approach is the development of a Cloud-Native Network orchestrator situated atop a multi-node cluster mesh infrastructure. This orchestrator is designed to facilitate flexible and dynamic container scheduling, providing a responsive and adaptable framework that addresses the unique considerations of healthcare data-sharing. By incorporating this orchestrator into our proposed framework, we intend to align dynamic elements of data-sharing use cases, associated policies, and infrastructure configurations, culminating in the seamless provisioning of SFC with minimal human intervention.

Recognizing the diversity inherent in digital health twin use cases, we acknowledge that the optimal deployment of Service Function Chaining varies based on the specific nature of each use case. To address this, we implement a tuning mechanism for hyperparameters, allowing for the prioritization of resource utilization or latency, thereby aligning with the performance requirements dictated by the unique characteristics of each use case.

This chapter presents an adaptive network orchestration approach tailored specifically for digital health twin use cases. Notably, our framework is characterized by its policy awareness, requirements awareness, and resource awareness, ensuring a holistic solution that navigates the intricate landscape of healthcare data-sharing. Through the convergence of adaptive VNFs, dynamic container scheduling, and fine-tuned hyperparameters, our proposed framework endeavours to set the stage for an efficient and responsive Digital Twin ecosystem in healthcare, overcoming the hurdles posed by policy constraints, network demands, and resource limitations.

7.2 Related work

SFC and VNF provisioning problems have been formalised and addressed by employing heuristics or DQL in the past. Recently, (118) formalised a heuristic-based approach to maximise the network throughput, while considering resource overhead. They consider the required CPU consumption of a VNF, link capacity, and maximum tolerable delay to search for optimal provisioning decisions. (114) takes a different approach and utilises DQL neural networks to outperform linear programming approaches in an effort to solve the SFC resource allocation problem. Similarly, the authors in (73) use RL-based techniques to formalise the same

problem as an MDP and address it with a policy gradient learning agent. We add to the current literature three main contributions: 1) We specifically reason about the type of SFC that is being provisioned to enforce network security policy in DHT use cases, 2) We propose heuristic-boosted DQL techniques to guide and facilitate the learning process according to prior knowledge of profiled data, 3) We add dynamic constraints to the provisioning problem; dynamic N-PoP candidate and prioritising different metrics with different use cases.

7.3 Health Data sharing Policies

In previous work (65), we defined the collaboration logic model which the EPI data-sharing framework follows to aggregate higher-level data-sharing agreements with lower-level network security goals to establish a policy-abiding data-sharing session. This can be further translated and aggregated to be enforced at a lower level. The policy additionally depends on the parties involved, and the data type being shared. After discussions with the hospitals within the EPI consortium, we can enumerate the following security goals:

1) Providing access control to the data resources,

2) Identifying and authenticating parties,

3) Health data integrity, and confidentiality,

4) and non-repudiation.

These security goals can be achieved by applying different security mechanisms, such as deploying access control and security protocols namely, SSL, SSH, IPSec, firewalling, and the security gateway systems (75). We take a separation of concerns approach where we define two levels of policies: data level, and network level. We assume that all policies fall under one of the two levels, and this is further discussed in (37).

By virtualising the network services, we aim to deploy and provision on-the-fly exemplary reliable VNF, which we call BFs, that can accomplish these security goals. The framework's goal is to define an adaptive BFC orchestrator, enforcing all types of network policies. We provide an access control mechanism by containerising a ready-to-deploy firewall function, and we address the rest by implementing standard security protocols to encrypt traffic. Once the security goal is specified, then we can map that to the mandated network services, and to the defined enforcement primitives: *Filter* traffic (**F**) and/or *transform* traffic (**T**).

The framework dynamically provisions these services by placing the BFs on available N-PoPs (Network Points of Placements), assigning the service requests to the running function, and routing traffic along the function's chain to enforce a policy. Along with the data-sharing policy changes, the available N-PoPs are

constrained with different use cases running. As a result, the high-level policy is defined and specified by two 3-tuples: $< actors, acts, inRelation >$, and $< endNodes, BFC, N\text{-}PoPs >$, where the second one is in accordance with the lower-level network policy.

7.4 DHT use cases and N-PoP restrictions

The three use cases, as previously defined, describe the requirements, restrictions, and network configurations associated with each one.

7.4.1 Electronic Health Records Repository

This use case is built to run EHR (Electronic Health Records) data-sharing scenarios where there are two N-PoPs affiliated with healthcare institutions (HI), and an isolated third-party research centre N-PoP. Fig. 7.1 illustrates the use cases' configuration and the infrastructure setup, where the data-sharing movement is expected from a remote user network to the HI network and vice versa. This use case requires remote access to sets and effective queries (Update/Insert/Get) of a patient's medical records.

The third-party research centre is uninvolved in this transaction, and hence the affiliated third N-PoP is isolated from the rest. The placement algorithm will not consider it while placing the BFC request and will place the functions on the other two to secure network traffic according to the BFC request.

7.4.2 Machine Learning model sharing

In an effort to advance healthcare research, we need to accelerate and support the deployment of ML-featured applications (such as psychiatry diagnosis, effective drug prescriptions, and side effects predictions, etc.), we define this use case where ML and analytics algorithms are sent from a research centre to be trained on data residing in the HI. Moreover, data movement is allowed again from the data provider (the HI) back to the algorithm provider (the research centre), so that the distributively trained model can be joined back again into one (more accurate) model, and shipped back to be ready to use.

As a result, the policy dictates the availability of links across affiliated N-PoPs to ensure who and where data is handled, as illustrated in Fig. 7.2. After establishing those restrictions on traffic flow, we then reconfigure the network to adapt to that and secure the network traffic.

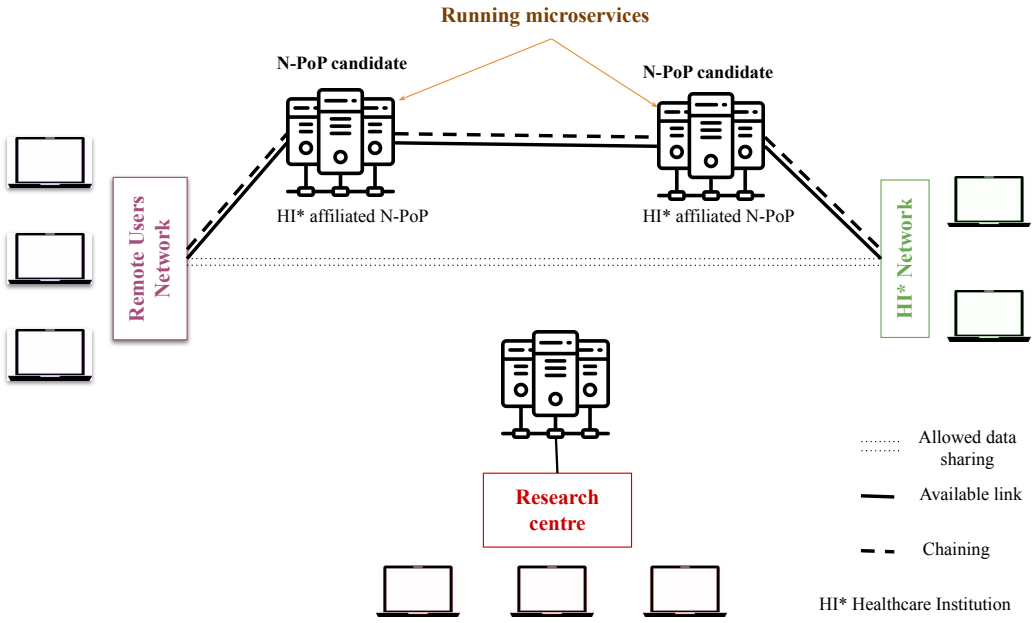


Figure 7.1: The infrastructure graph configuration under the EHR and the Streaming use cases.

7.4.3 Healthcare data streaming

This use case describes health data streaming, an example application is to monitor the patient's status via wearable data, provide timely interventions, etc. The data involved in this use case is sensitive data, which means that similarly as in use case A, HI-affiliated N-PoPs are only considered for placement.

7.5 BFC provisioning

7.5.1 Adaptive Provisioning of BFC

We need to adapt the network to performance and policy requirements by automating the provisioning of the network service function chains. To do that, we consider multiple approaches to provide a best-effort placement of network microservices on a Kubernetes cluster mesh of N-PoPs middleboxes. We consider the following provisioning steps:

1) Profiling of BFC service chains: Determine the computing and network profiles while running microservices within the service chain under different use cases.

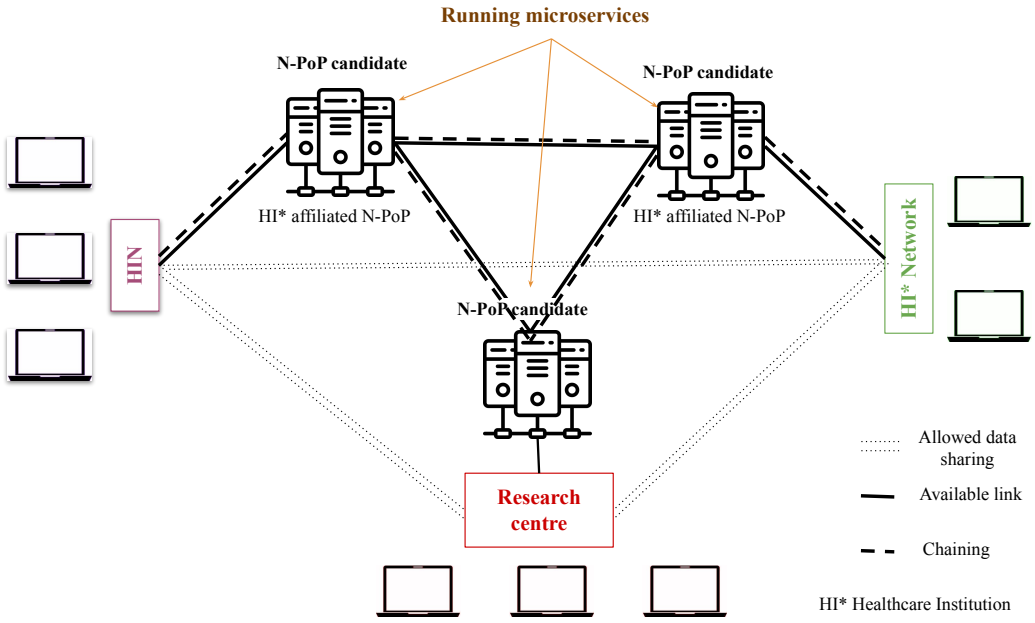


Figure 7.2: The infrastructure graph configuration under the ML model sharing use case.

2) Mapping BFC requests to running microservices: Assign new requests to run microservices' deployments under different constraints.

3) Allocation of N-PoP: When mapping fails, place a new instance on an N-PoP. This is also done according to a set of placement rules/constraints.

4) Chaining the microservices: This step is to assign available (routable) links and host virtual links and actually chain the microservices according to internal packet flow requirements between the microservices pairs.

7.5.2 Provisioning Decisions model

Infrastructure

The currently available infrastructure is represented as a directed, attributed graph $G(C, L, CPU^C, D^L)$. The finite set of vertices C represents the clusters that serve as network points of placements (N-PoP), where a network microservice can be placed. The unidirectional link between clusters is represented by the set $L \subseteq C \times C$, where $(i, j) \in L$ represents the unidirectional link between cluster c_i and $c_j \in C$.

Within the infrastructure environment, there exist limited computing and network resources. These are represented by the attribute sets CPU^C and D^L.

${cpu^C}_i \in CPU^C$ is the maximum CPU capacity of cluster $c_i \in C$, and ${d^L}_{(i,j)} \in D^L$ is the network capacity with an associated delay of link $(i,j) \in L$ at time t.

The configuration of G is representative of the data-sharing policy held by the infrastructure providers within the EPI consortium, such that for any cluster c_i there is a cluster c_j considered as an N-PoP candidate under the condition that:

$$(i,j) = \begin{cases} 1|exists, & \iff c_j \text{ is considered N-PoP candidate} \\ 0, & otherwise. \end{cases} \tag{7.1}$$

The lack of an edge would drive the placement algorithm to either centralise placement on c_i or distribute it on a different cluster c_z such that (i,z) exists.

As an example, when data provider A and algorithm provider B are sharing data, the policy dictates that a third-party associated N-PoP should not be considered for placement.

BFC requests and profiles

BFCs are ordered sets of network service requests, and a single request can be composed of multiple chained microservices. To successfully provision these requests, microservices can be placed and hosted on one or more clusters. $BFC = \{f_1, ..., f_n\}$ is a first-come-first-placed ordered set, where to each request $f \in BFC$ a directed graph $G_f = (S_f, L_f)$ is associated. Service requests are allowed to run as long as required, and multiple service requests can populate the infrastructure at any time. The finite set S_f represents the microservice functions that need to be assigned to run a request $f \in BFC$.

Moreover, $S_f \subseteq \mu S$, the list of all possible containerised network microservices (*e.g.* firewall, encryption, load balancer, NAT, etc.). The edges of the graph are members of the set $L_f \subseteq S_f \times S_f$ and are associated with inter-virtual links between two microservices, where $(q,r) \in L_f$ represents the unidirectional link between requested microservices μs_q and $\mu s_r \in S_f$.

Members of μS can be instantiated multiple times and can be deployed as a microservice replica with different configurations, such that ${\mu s_m}^n$ is the n^{th} instance of microservice $\mu s_m \in \mu S$. With that, one can instantiate smaller and bigger instances of the same microservice.

To model this, we define $CPU\{{\mu s_m}^n\}$ and $d\{{\mu s_m}^n\}$, the computing capacity and processing delay of the n^{th} instance of microservice μs_m. On the other hand, placed and running microservices can be shared across multiple incoming network service requests, as an example, ${\mu s_m}^n$ can be active for f_1 and f_2.

A network service request has profiled requirements values that may differ under different use cases utilising the requested service chain setup. As an example, a service request f active for a streaming use case needs more CPU resources and higher bandwidth than active for another use case. To model this, we define

$CPU^u_{q,f}$, D^u_f, representing the required CPU running microservice μs_q when active for f under use case $u \in U$, the end-to-end maximum delay respectively. (U is the set of all possible use cases)

7.5.3 Variables

The goal of the proposed placement algorithm is to allocate network microservice instances to N-PoP clusters and then map the deployed microservices to the running instance. Moreover, the algorithm considers the use cases being deployed over the network infrastructure and tailors the placement according to profiled resource consumption per use case. Furthermore, the microservices need to be chained to enforce the network policy route.

The first variable we consider is the virtual link mapping function $M^C_{(i,j),(q,r),f}$, where we decide to utilise link (i,j) with virtual link (q,r) under request f such that:

$$\mathrm{M}^C_{(i,j),(q,r),f} = \begin{cases} 1, & \text{link } (i,j) \in L \text{ is mapped to link } (q,r) \in L_f \\ 0, & \text{otherwise.} \end{cases} \tag{7.2}$$

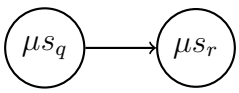
Then we consider is the placement function $P^C_{c_i,\mu s^n_m} \in \{0,1\}$, such that:

$$P^C_{c_i,{\mu s_m}^n} = \begin{cases} 1, & {\mu s_m}^n \text{ is placed on } c_i \in C \\ 0, & otherwise. \end{cases} \tag{7.3}$$

We also consider the mapping function $M_{\mu s^n_m,q,f} \in \{0,1\}$ to assign a microservice $\mu s_m \in \mu S$ needed by the service request f, and associated with $\mu s_q \in S_f$, to an instance ${\mu s_m}^n$ running on the cluster c_i, such that:

$$M_{\mu s^n_m,q,f} = \begin{cases} 1, & \text{microservice } {\mu s_m}^n \text{ is mapped to } \mu s_q \\ 0, & otherwise. \end{cases} \tag{7.4}$$

Note that variables defined in *Eq.* 7.5.3, 7.3, 7.4 are dependant values such that, suppose that we have a subgraph of S_f:



In fact, the possibility of connecting microservices depends on their mapping and placement. For example, we can have a successful microservice provisioning of μs_q can be that: $M_{\mu s^n_m,q,f} = 1$ and $P^C_{c_i,\mu s^n_m} = 1$. While the output of provisioning μs_r such that $M_{\mu s^p_k,r,f} = 1$, and $P^C_{c_j,\mu s^p_k} = 1$ is conditional to the existence of the link (i,j). This means that virtual link mapping $M^C_{(i,j),(q,r),f} = 1$, and it is determined, such that:

$$\mathrm{M}^{C}_{(i,j),(q,r),f} = 1 \implies \exists M_{\mu s^n_m,q,f} = 1, P^{C}_{c_i,\mu s^n_m} = 1 \wedge \exists M_{\mu s^k_p,r,f} = 1 P^{C}_{c_j,\mu s^k_p} = 1$$

(7.5)

Before assigning a microservice of flow request f to cluster c_i, we define the available CPU resources of c_i as:

$$cpu^C_i - \sum_{\forall \mu s^n_m | \mu s_m \in \mu S} P^{C}_{c_i,\mu s^n_m}.CPU\{\mu s^n_m\} \tag{7.6}$$

The expected latency on utilised links running flow requests f at time t is:

$$\hat{D}_{L,f} = \sum_{\forall (i,j) \in L, (q,r) \in L_f} M^{C}_{(i,j),(q,r),f}.d^{L}_{(i,j)}, \tag{7.7}$$

$$\hat{D}_{C,f} = \sum_{\substack{\forall c_i \in C, \\ \mu s^n_m | \mu s_m \in \mu S, \\ \mu s_q \in S_f}} M_{\mu s^n_m,q,f}.d\{\mu s^n_m\} \tag{7.8}$$

The un-utilised resources on already running microservice instances are approximated to be:

$$cpu^C_i - \sum_{f \in BFC} \sum_{\substack{\forall c_i \in C, \\ \mu s^n_m | \mu s_m \in \mu S, \\ \mu s_q \in S_f}} M_{\mu s^n_m,q,f}. \sum_{u \in U} CPU^u_{q,f} \tag{7.9}$$

7.5.4 Objectives and placement constraints

The objective is to minimise end-to-end latency and maximise CPU utilisation across the infrastructure's clusters, such that:

$$min \sum_{\substack{\forall c_i \in C, \\ \mu s^n_m | \mu s_m \in \mu S}} P^{C}_{c_i,\mu s^n_m}, \tag{7.10}$$

$$min(\hat{D}_{C,f} + \hat{D}_{L,f}) \leq D^u_f, \tag{7.11}$$

The two minima might not always correlate depending on the use case, hence the provisioning tools should prioritise minimising one over the other when appropriate.

7.5.5 Possible constraints

The first constraint is to ensure that the allocated function instances on the cluster c_i do not exceed available CPU resources, such that:

$$\forall c_i : \sum_{\forall \mu s_m^n | \mu s_m \in \mu S} P^C_{c_i, \mu s_m^n}.CPU\{\mu s_m^n\} \leq CPU_i^C \tag{7.12}$$

The second constraint is to ensure that the mapped microservice requested does not exceed the available CPU at the running instance:

$$\sum_{\substack{\mu s_q \in S_f, \\ f \in BFC}} M_{\mu s_m^n, q, f} \sum_{u \in U} CPU^u_{q,f} \leq CPU\{\mu s_m^n\} \tag{7.13}$$

Eq. 7.14 is to ensure that the maximal latency allowed is greater than the delay composed of link and processing delays.

$$\forall u : \sum_{\substack{\forall \mu s_m^n | \mu s_m \in \mu S, \\ f \in BFC, \\ \mu s_q \in S_f}} M_{\mu s_m^n, q, f}.D\{\mu s_m{}^n\} + \sum_{\substack{\forall (i,j) \in L, \\ f \in BFC, \\ (q,r) \in L_f}} M^C_{(i,j),(q,r),f}.D^L_{(i,j)} \leq D^u_f \tag{7.14}$$

If a cluster c_i is chosen for placement of n^{th} microservice of μs_m requested by f, the function needs to be running (placed) before assignment.

$$M_{\mu s_m^n, q, f} \leq P^C_{c_i, \mu s_m^n} \tag{7.15}$$

All microservices requested by request f are placed and mapped to the infrastructure.

$$\forall f \in BFC : \sum_{\substack{\forall \mu s_m^n | \mu s_m \in \mu S, \\ \mu s_q \in S_f}} M_{\mu s_m^n, q, f} = 1 \tag{7.16}$$

Building virtual paths over available links must obey the following rule to ensure that there exists a mapped link (i, j) to (q, r) if μs_q is placed on c_i and μs_r is placed on c_j, such that $\mu s_m^{n\prime}$ is potentially a different microservice instance assigned to μs_r:

$$\forall \mu s_q, \mu s_r \in S_f, (q,r) \in L_f : \\ M_{\mu s_m^n, q, f}.P^C_{c_i, \mu s_m^n}.M^C_{\mu s_{m\prime}^{n\prime}, r, f}.P_{c_j, \mu s_{m\prime}^{n\prime}} = M^C_{(i,j),(q,r),f} \tag{7.17}$$

Lastly, the microservice should not run indefinitely once instantiated, but the microservice is deleted after being idle for a specified duration of time T, such that $P^C_{c_i, \mu s_m^n} = 0 \iff \sum_{t=t'}^{t=t'+T} CPU\{\mu s_m^n\}$.

7.6 Provisioning Approaches

We deploy three approaches in an effort to satisfy the use cases' requirements: a greedy heuristic approach, Deep Q-Learning (DQL), and a Heuristic-boosted DQL (HDQL). Fig. 7.3 illustrates the high-level overview of the infrastructure orchestrator with the different components to make the provisioning decisions.

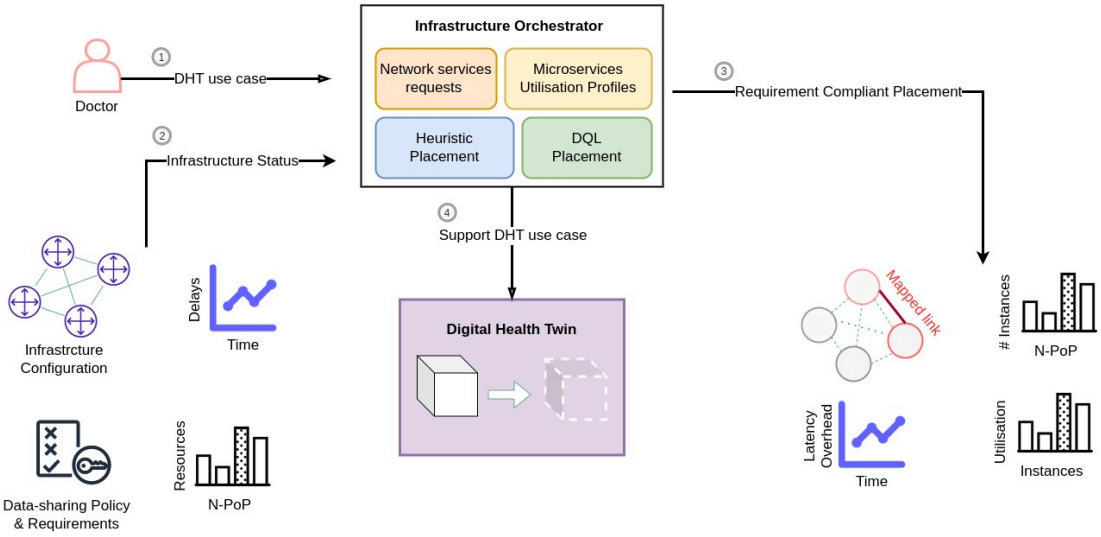


Figure 7.3: A high-level overview of the orchestrator.

We first deploy a greedy-based heuristic approach to reduce complexity with increasingly complex network policies and use cases and still provide a manageable best-effort provisioning decision by choosing the first N-PoP candidate meeting the placement constraints. This approach is dependent on accurate CPU profiles and does not react to resource usage and network latency bursts and anomalies. It is not concerned with the most optimal provisioning choice, instead, it provides the decisions that work.

7.6.1 Greedy Heuristic BFC Deployment Algorithm

This algorithm loops over the requested microservices and N-PoP candidates (lines 1-2), if a running microservice instance μs_m^n meets the requirements set according to placement properties and constraints in the previous section (line 4) then, μs_q is assigned to run on the cluster c_i and the loop continues to consider more microservice requests. If the assignment of a microservice fails on all clusters, then new instances need to be placed, and the algorithm loops again over clusters, but this time places new instances and then assigns them to a microservice request. The algorithm applies constraints defined in Eq. (12), (13), (14), (15), and (16) in lines 4 and 12, and it does so according to CPU and delay profiles. We only instantiate and place a new service when all the mapping fails

Algorithm 5 Heuristic BFC Deployment Algorithm

1: **procedure** PROVISION($G(C, L)$, $G_f(S_f, L_f)$)
2: **while** Not all μs_q has been mapped to $c_i \in C$ **do**
3: **while** Not all $c_i \in C$ has been checked **do**
4: **if** $c_i \in$ N-PoP candidates **then**
5: **if** μs_m^n meets requirements **then**
6: $M_{\mu s_m^n, q, f} = 1$
7: **end if**
8: **end if**
9: **end while**
10: **end while**
11: **if** Not all μs_q mapped to C **then**
12: **while** Not all $c_i \in C$ has been checked **do**
13: **if** $\mu s_m^{n'}$ meets requirements **then**
14: $P^C_{c_i, \mu s_m^{n'}} \wedge M_{\mu s_m^{n'}, q, f} = 1$
15: **end if**
16: **end while**
17: **end if**
18: **return** $P^C_{c_i, \mu s_m^n}, M_{\mu s_m^n, q, f}$
19: **end procedure**

(in line 13), and by that, we prioritise minimising placement, as in Eq. (10) under the constraints of delay.

7.6.2 DQL Algorithm

The performance of the first approach is highly dependent on the accuracy of the CPU and delay profiles. Compared to traditional heuristic-based resource scaling methods, Reinforcement Learning-based (RL) solutions are equipped to deal with un-profiled network and resource bursts, instead, this DQL approach relies on querying the current state and reacting via provisioning actions to maximise performance rewards.

Action space: The discrete actions of microservices placing and assignment are structured as [Cluster ID, Place/ Map/ Destroy, Instance ID, Microservice ID, Proxy ID]. The first value specifies the cluster that is considered; the N-PoP candidate. The second value specifies the type of provisioning action: placing a new instance, mapping a running instance to a request, or deleting an idle microservice instance. The third and fourth values, respectively, refer to the microservice instance and the type of microservice (n and m within μs_n^m). Lastly, the Proxy ID identifies the proxy we are configuring to chain the instantiated microservices and handles the network services requests.

State Space: The state space consists of the infrastructure's variables, which the DQL interacts with to monitor the environment's changes based on decisions. The provisioning is done across multiple sites, and the DQL algorithm needs to make a satisfactory trade-off between resource cost and latency. Therefore, the state of the agent should contain information about CPU utilisation/cluster, placement status of microservices, the number of microservices across the clusters, and the response time of requests.

Reward Function: The reward function measures the performance incentive for the agent to perform a new action, based on the infrastructure's current state. The entire reward R_{all} at time t is calculated as the weighted sum of reward in resource cost R_{res} and reward in performance R_{perf}, which is shown in:

$$R_{all} = \alpha R_{res} + \beta R_{perf} \tag{7.18}$$

Hyperparameters α and β are used to control the importance of these two values compared to the entire reward. One effective definition of reward function steers the agent towards better performance with higher utilisation of resources (prioritizing Eq. (10) vs Eq. (11)).

Action Policy: In the traditional DQL approach, the policy is set to map an observable state s_t to a provisioning action a_t at a time t. The policy is optimised by learning the Q-value performing a_t in state s_t, according to the following formula:

$$\pi(s_t) = \begin{cases} max_{a_t} Q(s_t, a_t), & if q \leq p \\ a_{random}, & otherwise \end{cases} \tag{7.19}$$

, where q is a random value with uniform probability in $[0, 1]$, and $0 \leq p \leq 1$ is the exploration/exploitation ratio parameter.

7.6.3 Heuristic-boosted Algorithm

While the previous tool can reactively adapt the network's configuration to optimally provision BFC requests, it still can take a long time to converge with increasingly complex requests, use cases, and N-PoP configurations. Subsequently, we propose to combine both provision approaches (A and B) to deploy a Heuristic-boosted DQL provisioning of SFC. We aim to accelerate the decision-making, and guide the model learning via a new action policy:

$$\pi(s_t) = \begin{cases} argmax_{a_t}[Q(s_t, a_t) + H(s_t, a_t)], & if q \leq p \\ a_{random}, & otherwise \end{cases} \tag{7.20}$$

Where the H-value $H(s_t, a_t)$ influences the choice of action, by evaluating the importance of executing the action a_t (suggested by the heuristic algorithm) having state s_t ([23]).

7.7 Experiments and Results

Use cases' Network Policies: The security enforcement primitives are consistent throughout all the use cases, such that the *firewall* function is mandated to provide access control to incoming traffic towards the healthcare institution. Likewise, all outgoing traffic should be *encrypted* to protect sensitive data. This is showcased in Fig. 7.4, where you can see the same logic being applied to all the use cases.

The EHR use case is profiled to be a relatively small load application with an expected average send rate of 100-200 kB/s. The ML-model sharing use case is defined to be, also, a small load use case with an expected (average) periodic send rate of 100-200 kB/s. Additionally, the streaming use case is profiled to be a large load use case with a 1-3 MB/s send rate. Accordingly, the CPU profiles of different BFCs under different use cases' send rates are recorded in (62), and further used via the heuristic algorithm to make provisioning decisions.

Moreover, the provisioning algorithm should comply with multiple use case requirements. Maximum CPU usage is prioritised with the EHR and ML model sharing use cases, compared to the streaming use case where low latency overhead is required as well.

7.7.1 Experiments

To evaluate placement and assignment decisions taken by the different provisioning tools, we run the three use cases according to the defined send rate workloads, and associated CPU profiles. We first reconfigure the graph G as Kubernetes clusters, to reflect Fig. 1 and 4. Connected clusters form a cluster mesh via a cilium backend server [1].

New instances vs CPU utilisation: We increase the number of concurrent clients utilising the BFC with locust[2], and we record the instantiated new instances/microservice across all the clusters under the different use case configurations, with the growing number of clients. For replication purposes, the implementation and configurations of the experiments are available on GitHub [3]. Furthermore, we compare the number of pods (instances) to the actual utilized CPU per instance, to evaluate the CPU resource wastage, and further relate that to the latency.

Latency Overhead: Similarly, we increase the number of concurrent clients running different use cases and record the latency of processing one request (sending a request and receiving a reply back). Low latency can be accomplished by providing high-performance networking, but the goal is to evaluate the overhead latency caused by adding network services in between end nodes, and the effect

[1] https://cilium.io/
[2] https://locust.io/
[3] https://github.com/epi-project/Netsoft2023

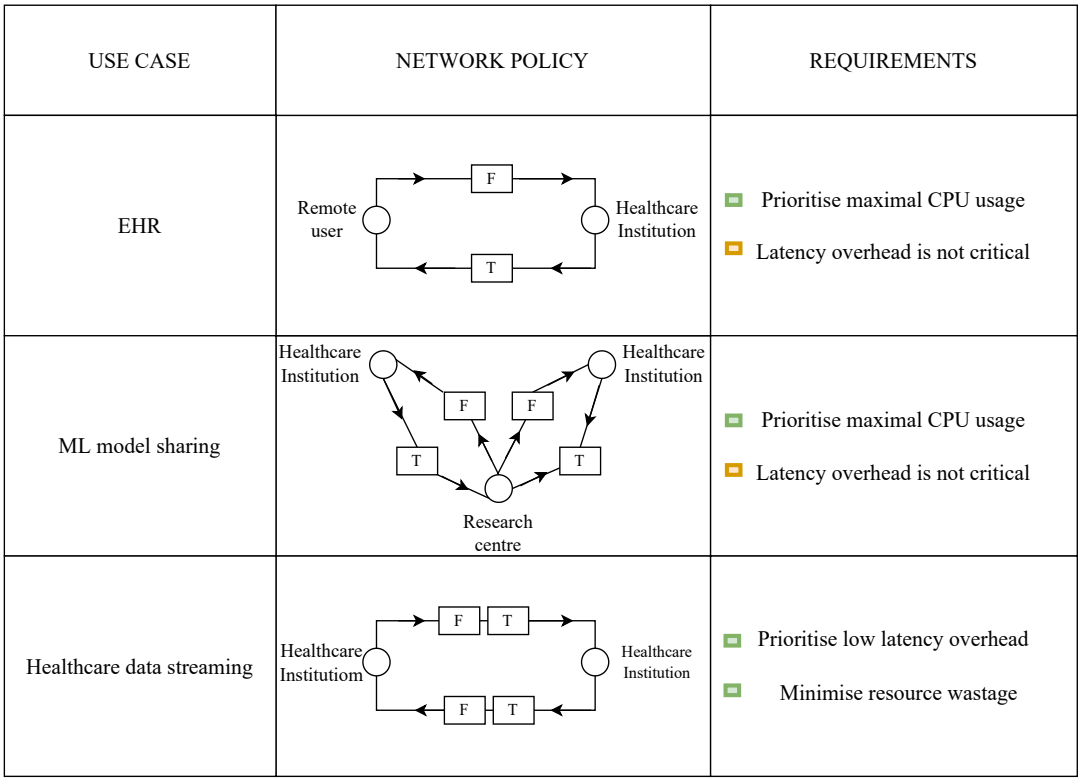
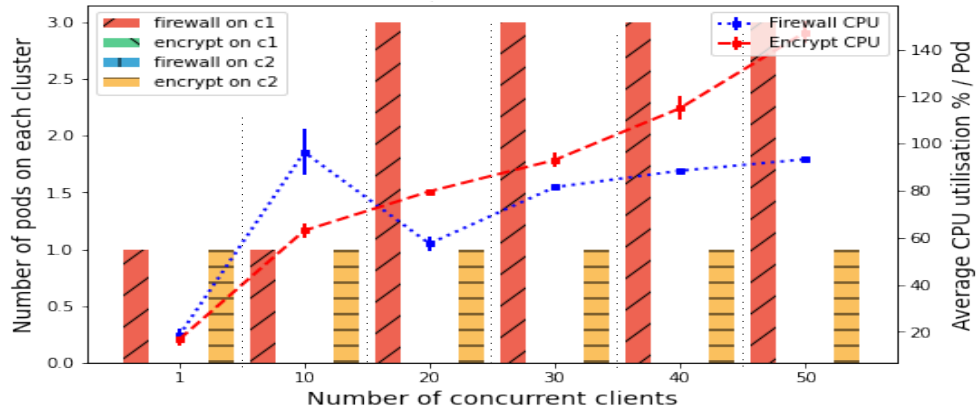


Figure 7.4: Different network policies and requirements, deploying different use cases; **F** represents a firewall microservice request, and **T** represents an encryption microservice request.

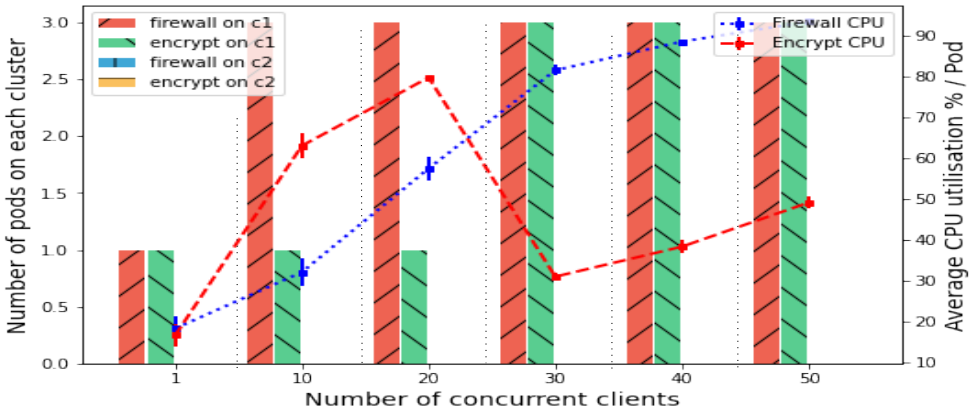
of different provisioning decisions. We try to minimise the delay when deploying the network function request, and ideally the latency overhead $\approx 0ms$. We collect the latency overhead by calculating the average of 10 queries and then measuring the effect of the chain addition compared to no functions in between.

7.7.2 Results

Under the first use case's N-PoPs constraints, no microservice instances are placed on cluster 3 *i.e.* the Research centre cluster (or, if already running, not assigned to the requests running EHR traffic), as shown in Fig. 7.5. The provisioning decision, however, differs with each method, to capture the difference we collect the number of running pods /clusters and the effect of increasing the number of clients from 1 to 50 concurrently running. Moreover, the provisioning decisions as we discussed previously are based on factors: CPU availability, and latency



(a) The heuristic algorithm placement

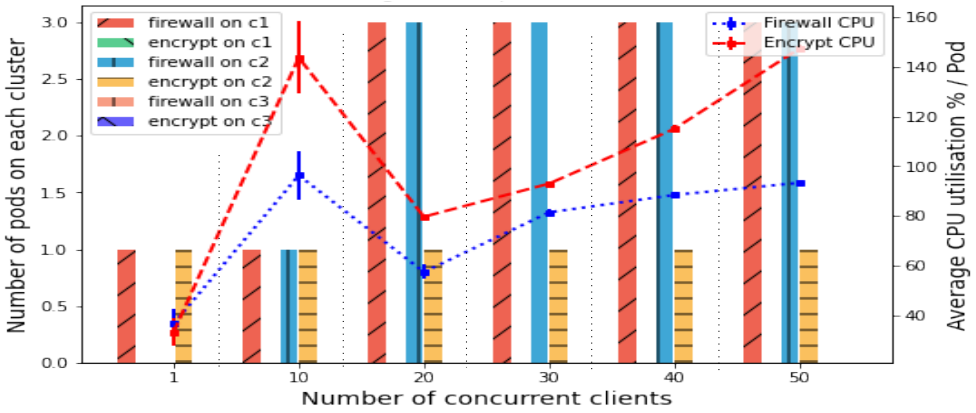


(b) The DQL and HDQL algorithm placement

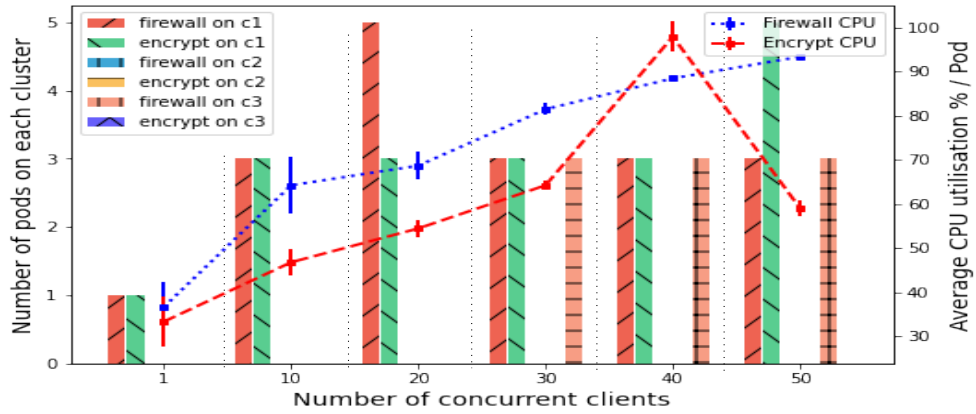
Figure 7.5: The placement of microservices and the average CPU utilisation running the EHR use case.

overhead. Hence, we also showcase the average CPU utilisation percentage/pod with each iteration, and the latency recorded to successfully resolve one request.

Firstly, in Fig. 7.5(a) we notice that the heuristic-based placement starts by placing small configured firewall and encryption microservices on cluster 1 and 2, respectively, with the CPU utilisation average of 20-25% running 1 client on both. The firewall utilisation ramps up faster than encryption to reach 100% with 10 concurrent clients, then the heuristic placement reacts to the maximal utilisation by upgrading the configuration of the microservice to reach 3 pods running on cluster 1 with 20 concurrent clients running. The trend we notice throughout this plot is that the heuristic placement is more concerned with maximising the average CPU utilisation, such that we end up overcommitting the firewall microservice and assigning it to the maximal ammount of traffic. The effect of overcommitting the firewall microservices (120%, 140% with 40 and 50 clients) is also reflected in the latency in Fig. 7.8(a), such that latency overhead



(a) The heuristic algorithm placement



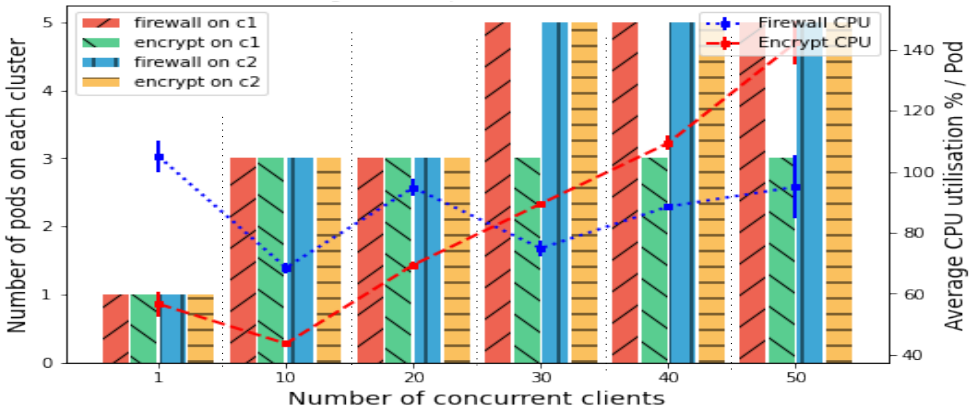
(b) The DQL and HDQL algorithm placement

Figure 7.6: The placement of microservices and the average CPU utilisation running the ML model sharing use case.

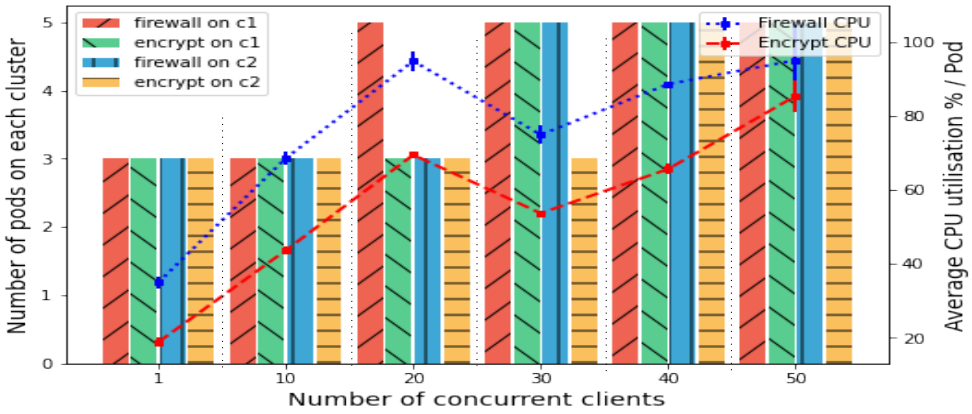
increases from 2.21 ms with 30 clients to 5.32 and 6.26ms with 40 and 50 clients, respectively.

On the other hand, with the DQL approach, this latency increase is avoided because the resources are never overcommitted, with the hyperparameters in Eq. 7.18 set so that $\alpha > \beta$, so the model will prioritise resources. With this approach, two main decisions to optimise placement are taken, the first is to place the encryption microservice on cluster 1, and by that only using the Kubernetes backend discovery service once, such that microservices on the same cluster belong to the same private network. Another thing is that the DQL is more adaptive to resource bursts, with no need for profiling, and relying on accurate profiles of CPU usage. This is also reflected in the latency, such that the overhead is halved compared to the heuristic approach, and it decreased from 5.32 to 1.26ms running 40 clients.

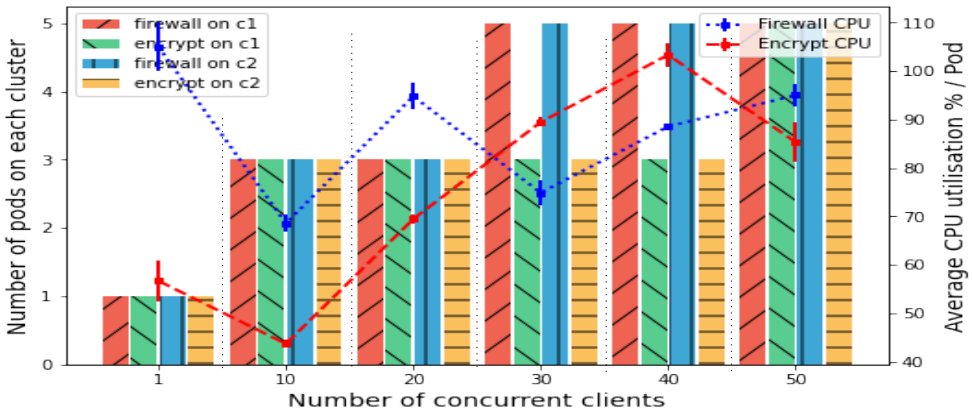
With the HDQL approach, there are no major differences, and we end up



(a) The heuristic algorithm placement



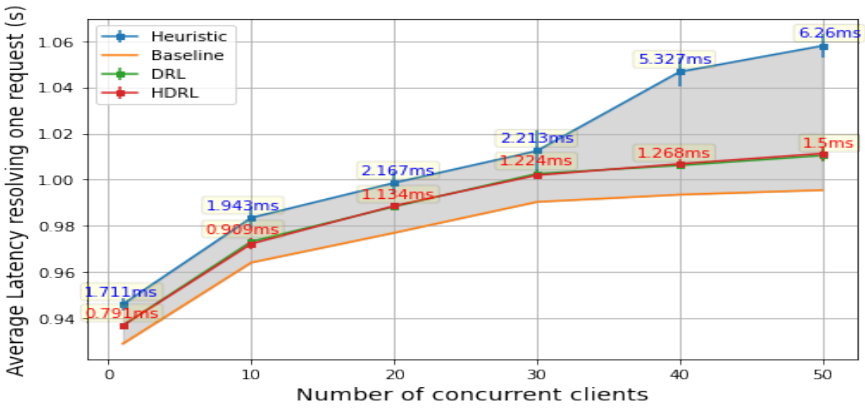
(b) The DQL algorithm placement



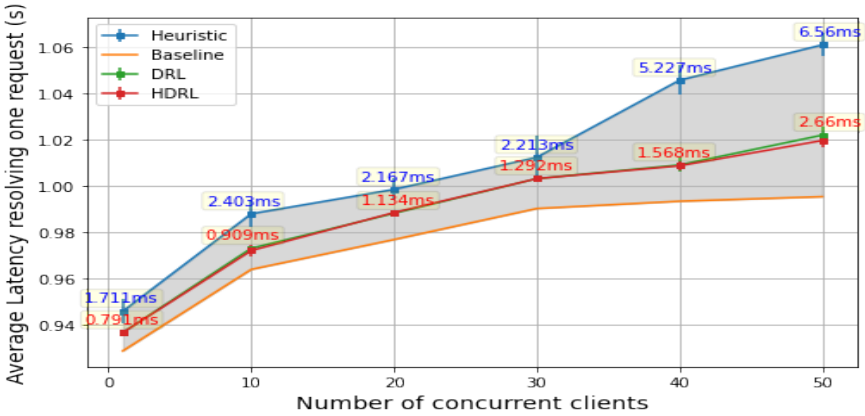
(c) The HDQL algorithm placement

Figure 7.7: The placement of microservices and the average CPU utilisation running the streaming use case.

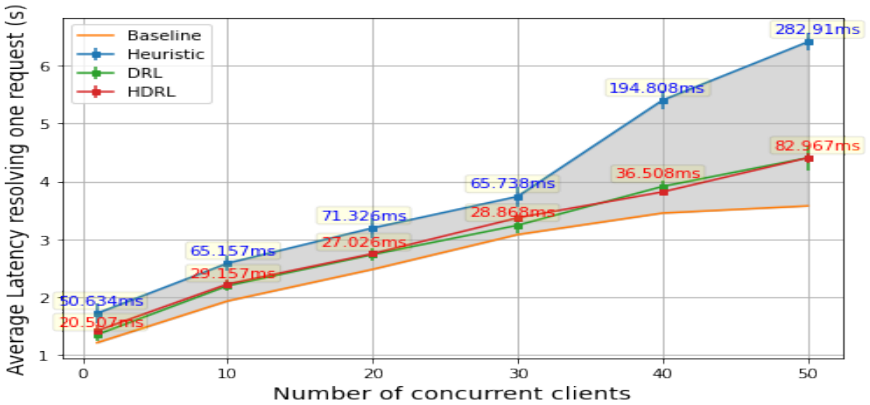
with the same latency as well. The reason is that the hyperparameters are set



(a) The EHR use case.



(b) The ML model sharing use case.



(c) The streaming use case

Figure 7.8: The latency average recorded with different placement methods, and the overhead compared to proxying traffic without passing through extra microservices.

in the action policy. With this use case, latency overhead is not crucial, and hence occasional under-provisioning of resources is not an issue, and the heuristic placement is a sufficient tool for this use case.

Similarly, the ML model sharing use case has initially a similar placement, the difference is that the utilisation is doubled indicating that the pods are assigned to handle double the requests. After that, we start another instance of a firewall on cluster 2, to handle half of the traffic. The heuristic function upgrades the placement configuration to adapt to the increasing workload. Similarly, this approach ends up overcommitting resources and the overhead increases as shown in Fig. 7.8(b).

The optimisation decisions taken by the DQL and HDQL approaches are demonstrated by deciding to place microservices on the same cluster, to then start new instances of the firewall cluster 3, that has currently associated with lower round-trip latency, compared to cluster 2. This approach adapts to resource usage bursts more accurately and minimises latency overhead accordingly.

With the last use case, the healthcare data streaming use case, we prioritise latency, and there are three different placement decisions taken. First, the heuristic performs poorly with this use case as shown in Fig. 7.8(c), where the chaining is still distributed, and assignment decisions require two different lookups. Unlike the other two approaches, the placement is distributed across clusters, but the chaining is on one cluster. With that, Kubernetes backend microservice discovery is optimised. With this experiment, due to the hyperparameters change, we end up wasting resources with the DQL approach as shown in Fig. 7.7(b). Although with these provisioning decisions, we provide the best latency overhead, there exist better actions and hence we have a third different placement with HDQL. The HDQL provisioning provides approximately equal latency with maximal CPU utilisation (shown in Fig. 7.7(c) and 7.8(c)).

As a result, the heuristic-based approach proved to be sufficient with the EHR and ML model-sharing use cases, where we end up under-provisioning but inflicting tolerable latency. That is especially true since low latency overhead is not crucial in running said use cases, instead, we prioritise minimal CPU wastage. The HDQL tool performs the best while running the streaming use case, where minimal overhead latency was achieved, but with seemingly no over-provisioning and resource wastage.

7.8 Conclusion

To run DHT use cases, it is essential to first consider data-sharing policies (including network policies), and translate them into actionable service function chain requests. Additionally, the provisioning of BFC (new instance placement and/or assignment of an incoming request to an old running instance) depends on the use case's requirements and the current state of the infrastructure. We need to

prioritize latency and/or minimise resource wastage, and we do that by modelling the decision as a constrained optimization problem. Initially, we addressed the requirements and constraints via heuristic, which can query the infrastructure state and output decisions influenced by the BFC CPU profiles. Next, we proposed using DQL methods, which provide more resilience to un-profiled bursts and network degradation. Finally, we combined both approaches to introduce a "best of both worlds" solution, that proved to be most valuable in accomplishing lower latency overhead, with minimal CPU wastage.

The provisioning tools should consider a combination of constraints and objectives. The framework we proposed can be used within any general context, and it effectively provisions network resources to deploy DHT use cases. We conclude that heuristic-based approaches are sufficient when the latency overhead is not crucial, while HDQL tools are most effective otherwise. In our next chapter, we evaluate the framework according to security and privacy risks, whilst running different workflows. EPI users compose workflows with multiple events including data in action, data in transition, and data in storage. Different workflows might entail a list of resources requisite; data type, dataset size, data sensitivity, and minimum data utility.

Chapter 8

Privacy by Design

Due to the inherited sensitivity of health data, institutions are still wary of sharing their data, especially with the increasing number of breaches in recent years and the strict privacy legislation involved (GDPR, HIPAA, etc.). There exist privacy and security concerns that arise with making data available for use or processing. To tackle these concerns, we incorporate Privacy by Design (PbD) principles. This informs our approach to constructing a data-sharing framework that aligns with said principles. Subsequently, we introduce examples of data-centric use cases requiring processing, followed by the delineation of the computation events model and data properties intrinsic to a use case. Furthermore, in order to gain insight into the potential privacy risks associated with executing a workflow request, we expand upon the model to quantitatively evaluate these risks. Subsequently, we construct a framework, the EPI framework, aimed at mitigating these identified risks, via adhering to PbD properties and provisioning extra services. In this chapter, we answer the question:

RQ4: "How to orchestrate the EPI framework services to minimise security and privacy risks by comprehensively modelling security/ privacy probabilities and mitigating risks of DHT data-sharing workflows according to privacy-defined attributes?"

This chapter is based on:

- **Jamila Alsayed Kassem**, Tim Müller, Christopher A. Esterhuyse, Milen G. Kebede, Anwar Osseyran, Paola Grosso, The EPI framework: A data privacy by design framework to support healthcare use cases, Future Generation Computer Systems, 2024, 107550, ISSN 0167-739X, https://doi.org/10.1016/j.future.2024.107550.Jamila Alsayed Kassem, Tim Müller, Christopher A. Esterhuyse, Milen G. Kebede, Anwar Osseyran, Paola Grosso, The EPI framework: A data privacy by design framework to support healthcare use cases, Future Generation Computer Systems, 2024, 107550, ISSN 0167-739X, https://doi.org/10.1016/j.future.2024.107550.

8.1 Introduction

In the era of big data and increased digitalization, the sharing of sensitive information, particularly in the medical domain, has become crucial for advancing healthcare research, treatment, and patient outcomes (53). However, the inherent privacy risks associated with sharing such data pose significant challenges. Traditional data-sharing approaches often neglect essential privacy considerations, potentially compromising data confidentiality, integrity, and control (44).

To address these concerns, the EPI framework gives control over integrating PbD principles into the sharing session. The EPI framework aims to establish a comprehensive and robust environment for sharing medical data whilst ensuring privacy and data utility. By incorporating workflow orchestration, data sharing policies reasoning, and security/ privacy as a service orchestration, EPI addresses the complex privacy challenges inherent in medical data sharing. The EPI framework provides a holistic approach to data sharing by considering multiple dimensions of privacy risks. These dimensions include Linkability (L), Identifiability (I), Non-repudiation (Nr), Detectability (Dt), information Disclosure (iD), data Indulgence (In), and policy Non-compliance (Nc). By thoroughly assessing these dimensions, EPI enables the impact and likelihood quantification of potential privacy risks associated with shared medical data and provides the means to mitigate this risk.

The EPI Framework focuses on maintaining data control throughout the data-sharing process, ensuring that privacy considerations are integrated from the outset. The framework addresses the challenge of balancing data utility with privacy protection by offering mitigations specific to the associated data risks identified during workflow submission.

This chapter elaborates upon prior research discussed in Section 8.2, with a particular emphasis on the recurring theme of PbD. We define PbD concepts in

section 8.3, and, in this chapter, distinguish between privacy and security properties. The use cases investigated in the EPI project are introduced in section 8.4, accompanied by explicit definitions of data properties and the events computation model associated with a use case (sections 8.5.1 8.6, respectively). We present the LINDDIN privacy assessment model in section 8.7, wherein risk attributes are formulated to quantitatively assess risk, considering both data properties and the computation model. Offering an overview of the EPI framework in 8.8, we delve into more detailed descriptions of the EPI orchestration layers (sections 8.9 and 8.10). Lastly, to evaluate the framework, we provide a detailed walk-through of a practical use case and quantify the extent to which privacy risks are alleviated before and after the implementation of the EPI framework in 8.11.

8.2 Related work

Several papers have contributed to the definition and development of PbD principles, frameworks, and applications. In (20), the authors outline the seven foundational principles of PbD, providing a framework for the operation of systems, processes, and technologies: proactive and preventive, default privacy, embedded in the design, full functionality of services, end-to-end security, visibility and transparency, and finally, respecting user privacy.

(76) introduces the concept of privacy-aware ubiquitous systems and presents principles for incorporating privacy by design into the development of such systems. It emphasizes the need to consider privacy throughout the design process. (92) proposes a PbD framework to evaluate the privacy gap between IoT devices and middleware platforms, such as OpenIoT, Eclipse SmartHome, etc. Minimising data, in general, is a common practice in Literature. As an example, in (44), the framework is based on minimizing data storage (retention), raw data intake, data sources, and data knowledge discovery.

In the context of PbD in healthcare applications, (21) and (97) apply the 7 PbD properties introduced in (20) to evaluate current practices; namely IoT applications. In the true spirit of PbD, (32) utilises deep learning tools to pseudo-anonymise neuroimages using defacing, skull stripping, and face masks to preserve privacy and utility.

There already exist data-sharing platforms that align themselves with PbD principles in general. One of them is Mahiru (111), a decentralised data-sharing platform that emphasises domain autonomy. A design choice that makes working with the framework practical is that it broadcasts all policy information to the site that at that moment acts as an orchestrator; however, in the medical domain, this is infeasible because some policies may be private.

The EPI Framework addresses the challenges of privacy and security in data workflows by utilizing three main components: *BRANE*, the *policy server*, and the *Bridging Functions Chain (BFC)* orchestrator. Through the utilization of

the EPI Framework, organizations can enhance privacy protection, promote data utility, and achieve compliance with privacy regulations. The framework's holistic approach aligns with the evolving landscape of privacy and security requirements, enabling organizations to adapt to changing data privacy landscapes and emerging threats.

8.3 Data Privacy by Design

PbD is a fundamental principle that advocates for embedding privacy protections into the design and development of systems. One prominent regulation that emphasizes privacy by design is the GDPR in the European Union. GDPR fixes strict requirements for data protection and privacy, placing increased responsibility on organizations to implement PbD principles.

Integrating privacy controls and safeguards into the processes may include implementing strong access controls, encryption mechanisms, pseudonymization and anonymization techniques, and user consent mechanisms. It also entails promoting transparency by informing individuals about the purposes and scope of data processing and ensuring that privacy settings are easily understandable to users. By incorporating PbD, organizations can establish a privacy-conscious culture, foster trust with users, and proactively mitigate privacy risks. PbD enables organizations to comply with legal requirements like GDPR and demonstrates a commitment to protecting individuals' privacy rights.

8.3.1 Privacy vs. Security properties

Security and privacy are closely related concepts but have distinct focuses. Security primarily deals with protecting systems, networks, and data from unauthorized access, breaches, and malicious activities. It encompasses measures such as authentication, encryption, firewalls, intrusion detection systems, and incident response mechanisms.

Privacy, on the other hand, is concerned with protecting personal information and ensuring that individuals have control over the collection, use, and disclosure of their data. It encompasses principles like data minimization, purpose limitation, consent, transparency, and individual rights. While security focuses on protecting the integrity, availability, and confidentiality of data and systems, privacy focuses on preserving individuals' autonomy, dignity, and control over their personal information. Both security and privacy are essential in today's digital landscape. Security mitigations might align with privacy needs. For example, protecting against unauthorized access ensures control over data disclosure, addressing concerns from both perspectives.

In system design, we need to recognize the importance of both security and privacy and adopt an integrated approach to protect data and individuals' privacy

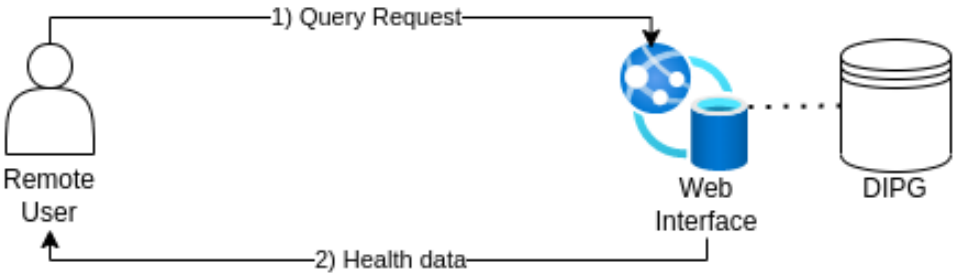


Figure 8.1: The workflow illustration of the DIPG use case.

rights. Balancing these aspects is crucial for building data-sharing frameworks, complying with regulations, and fostering a positive user experience in an increasingly data-driven world.

8.4 The DIPG-registry

In previous chapters, we defined our data-sharing use cases, and we will discuss these use cases in the context of PbD principles. These utilize patients-generated data within the EPI consortium.

One of the use cases under the EPI project is the Diffuse Intrinsic Pontine Glioma (DIPG) registry, founded by the SIOPE DIPG/DMG network. The registry was established in 2011 to advance DIPG research. The DIPG also known as diffuse midline glioma (DMG), is a rare paediatric brain cancer for which there is no curative treatment. The registry holds information on DIPG patients across Europe and a partner registry in North America, the International DIPG/DMG Registry, with patient data from the USA, Canada and Australia (15). The registry serves the goal of advancing DIPG research by granting researchers conditional access to selected datasets to perform analysis with more data points and encouraging members to contribute datasets to the registry.

The registry serves to improve DIPG research by granting members (conditional) access to selected datasets in order to perform analyses with more data points and thus higher efficacy. Members submit clinical data from their institution to the DIPG Registry via a secure online CRF-structured web application and database[1]. Figure 8.1 provides an illustration of the data flow required to run this use case. This is a simple workflow of query requests of remote users, and getting back the health data queried.

[1]https://dipgregistry.eu

8.5 Data properties

Medical workflows (pipelines) can involve various types of data with varying *sensitivity* and *utility*; usefulness of a data set to successfully run a specific workflow.

8.5.1 Data Sensitivity

Medical data is inherently sensitive, as categorized by the GDPR ([7]). However, this sensitivity is contingent on various factors. For instance, the state of the data—like whether it is encrypted—plays a pivotal role. Upcoming sections will revisit the specific data attributes that shed light on the associated privacy risks. We consider several data states within the EPI's data-sharing ecosystem, and are relevant to the data required for the three use cases:

- **Raw Data:** refers to data that is in its original, unprocessed format. It may include unstructured text, sensor readings, or any data that has not undergone any transformation or aggregation. Raw data often requires additional processing or anonymization to mitigate privacy risks before storage or sharing.

- **Anonymized Data:** data has undergone a process to remove or modify identifiers, making it extremely difficult or impossible to link the data back to an individual. Hence, the sensitivity of anonymized data is reduced. Anonymization techniques such as pseudonymization are applied to protect privacy.

- **Encrypted Data:** refers to data that is transformed using cryptographic algorithms into an unreadable format. Encrypted storage adds a layer of security by protecting the data from unauthorized access. The encryption helps mitigate the risk of exposure if the storage is compromised, or if a transfer session is sniffed (eavesdropping).

- **Synthetic Data:** artificially generated data that mimics the statistical properties of real data while containing no identifiable or sensitive information. Synthetic data is often used for testing, development, and research purposes. Since it does not contain real patient information, the sensitivity is typically low.

- **Aggregated Data:** data that has been combined and summarized from multiple sources or individual records. Aggregation helps protect privacy by reducing the granularity of the data and removing personally identifiable information. The sensitivity of aggregated data depends on the level of detail retained and the potential for re-identification.

Furthermore, with the data state in mind, we commonly think about the sensitivity of a current dataset based on the distance compared to the original (raw) dataset(s), as shown in equation 8.1, such that $sensitivity \in \{0, 0.5, 1.0\}$. Distance and sensitivity metrics are inversely proportionate, the higher the distance of a dataset is the lower the sensitivity is:

$$sensitivity = 1 - distance(raw, current), \tag{8.1}$$

where 1 is an indication of the most sensitive dataset, 0 is the least sensitive and $distance \in \{0, 0.5, 1.0\}$.

Moreover, the methods used to measure the distance metric are dependent on the dataset itself. As an example, statistical measures can be used to assess the difference in statistical properties, or if the dataset contains images, metrics like Structural Similarity Index (SSIM) or Peak Signal-to-Noise Ratio (PSNR) can be used to assess image distortion distance. The choice of a distance measurement method depends on the characteristics of the data and the specific modifications applied.

For simplicity purposes, let's assume that distance can be one of three values: 0 if data is not processed at all meaning $distance(raw, raw) = 0$, 0.5 if data is pseudo-anonymised, partially encrypted, partial data synthesis, or partially aggregated based on the state (privacy-preserving method). 1 if data is fully processed to maximize distance.

To provide an example, suppose we have a raw EHR dataset, as shown in listing 8.1, that contains patient records with sensitive attributes such as patient ID, age, gender, condition, date of visit, cholesterol level, and blood pressure. Each record in the dataset represents patients' visit history. Let's assume we have a simplified example of the dataset that is in Listing 8.1:

```
Raw = { "PatientID": 1, "Age": 42, "Gender": "Male",
  "Condition": "Diabetes", "Date_of_Visit": "2023-01-05",
  "Cholesterol_Level": 180, "Blood_Pressure": "120/80",
  "BMI": 25.5 }
```

Listing 8.1: An example, unprocessed EHR dataset

Listing 8.1 shows record *raw* in an unprocessed state, with no further processing to change the state. Hence $distance(raw, current) = 0$. In Listing 8.2, the data state is changed. In this pseudonymized dataset, the patient IDs have been replaced with pseudonyms "P1" through "P4". The date of the visit has been replaced with generic labels "Visit1" through "Visit4" to protect the privacy of individuals. The remaining attributes such as age, gender, condition, cholesterol level, blood pressure, and BMI are retained. The distance in this listing is considered to be $distance(raw, current) = 0.5$, under the assumption we previously stated.

```
Psuedoanonymised = { "PatientID": "P1", "Age": 42,
  "Gender": "Male", "Condition": "Diabetes",
```

```
3   "Date_of_Visit": "Visit1", "Cholesterol_Level": 180,
4   "Blood_Pressure": "120/80", "BMI": 25.5 }
```

Listing 8.2: An example simple pseudo-anonymisation of an EHR dataset

On top of what is omitted in the 8.2, the fully anonymized dataset in 8.3 also replaces the age attribute with "Age1" through "Age4", and the gender is anonymized as "Gender1" through "Gender4". The rest of the attributes related to medical conditions, such as condition, date of visit, cholesterol level, blood pressure, and BMI, remain intact with no clearly identifying information. The distance in the last listing is considered to be $distance(raw, current) = 1$.

```
1 Anonymised = { "PatientID": "Patient1", "Age": "Age1",
2   "Gender": "Gender1", "Condition": "Diabetes",
3   "Date_of_Visit": "Visit1", "Cholesterol_Level": 180,
4   "Blood_Pressure": "120/80", "BMI": 25.5 }
```

Listing 8.3: An example of an EHR dataset of an anonymised entry via a simple method

Bear in mind that while distance measurement offers a quantifiable evaluation of dataset disparities, it may not fully encapsulate the effect on data utility or usefulness. Hence, a comprehensive evaluation of altered datasets should account for both sensitivity and utility aspects.

8.5.2 Data Utility

Data utility refers to the value or usefulness of data for a specific purpose or task, such as the workflow associated with the three use cases. Defining all utility factors can be a cumbersome task, so for the sake of this paper, we focus our model on the factors that are most relevant to the provided use cases. The utility of data can be influenced by various factors, including data type, data state/distance, data size, and available computing resources.

Data type: In the context of health data, a multitude of data types may be necessary for the given use case, each subject to processing to deliver a specific service. For instance, data types encompass patients' health records, medical images, clinical trial data, sensor data, wearables, and the like.

Different types of data have different utilities depending on the specific algorithm and the problem being addressed. This is showcased when a data type u_{type} is available for processing or not when the workflow is requested:

$$u_{type} = \begin{cases} 1, & \text{type requested is available for workflow} \\ 0, & \text{otherwise.} \end{cases} \tag{8.2}$$

Computation Power: Handling and processing large, distorted, and synthetic datasets may require significant computational resources and can impact

the efficiency and scalability of; for example; the ML-based workflow (CVA use case). The utility of data is influenced by the available computing resources. If the computing resources are limited, the utility of large datasets may be reduced due to the constraints on processing and model training. Conversely, with ample computing resources, the utility of large datasets can be fully realized, allowing for more comprehensive analysis and better model performance. The compute resources relate to CPU and memory resources requested r_{req} compared to that available r_{av}, and is formalized as a flag:

$$u_{compute} = \begin{cases} 1, & r_{req} < r_{av} \\ 0, & \text{otherwise.} \end{cases} \tag{8.3}$$

Data State: The first two attributes, aforementioned, are functional attributes and depend on resources' availability, while the following two attributes affect the privacy risk directly. The status of the data and the relative distance can also impact its utility. Fully Synthetic data, which is artificially generated to mimic real data, may have lower utility compared to raw or real-world data because it may not capture all the nuances and complexities present in the original data. To capture this effect, we look into the distance value and compare it to the acceptable distance submitted along the workflow, to successfully run it. This distance attribute is formalized as a flag $u_{distance}$:

$$u_{distance} = \begin{cases} 1, & distance(raw, current) \leq distance_{req} \\ 0, & \text{otherwise.} \end{cases} \tag{8.4}$$

Data size: The size of the data can affect its utility in different ways. Larger datasets generally provide more utility by enabling more accurate and robust machine learning.

On the other hand, data minimization is a sign of good privacy by design practices, and one way to do that is by sharing (or making available) the minimal dataset size while running a workflow. Other than minimizing sensitivity (increasing distance), size is an attribute that needs to be considered under privacy and utility constraints.

The effect of the size attribute is formalized as flag u_{size}, where $u_{size} = 1$ means that this size utility condition is met:

$$u_{size} = \begin{cases} 1, & \text{size requested is available for workflow} \\ 0, & \text{otherwise.} \end{cases} \tag{8.5}$$

The requested utility attributes are set by the users submitting a workflow request, and the current availability of data and computing resources are determined by the framework. As a result, the total utility (U) is calculated as the sum of these attributes, such that α, β, θ, and γ parameters are also set to

determine the relevance of these attributes to the submitted workflow. Where $\alpha + \gamma + \beta + \theta \leq 1$:

$$U = \alpha u_{type} + \gamma u_{compute} + \beta u_{distance} + \theta u_{size} \tag{8.6}$$

As mentioned before, data privacy-preserving methods are associated with data sensitivity, as well as data size. Figure 8.2 showcases the effect of these methods on utility. The framework we will define aims to find the optimal trade-off point to run a workflow, to minimize privacy risk with utility as a constraint. The threshold for data privacy is set by the data sharing policies, while the threshold for minimum utility is set by the user submitting the workflow.

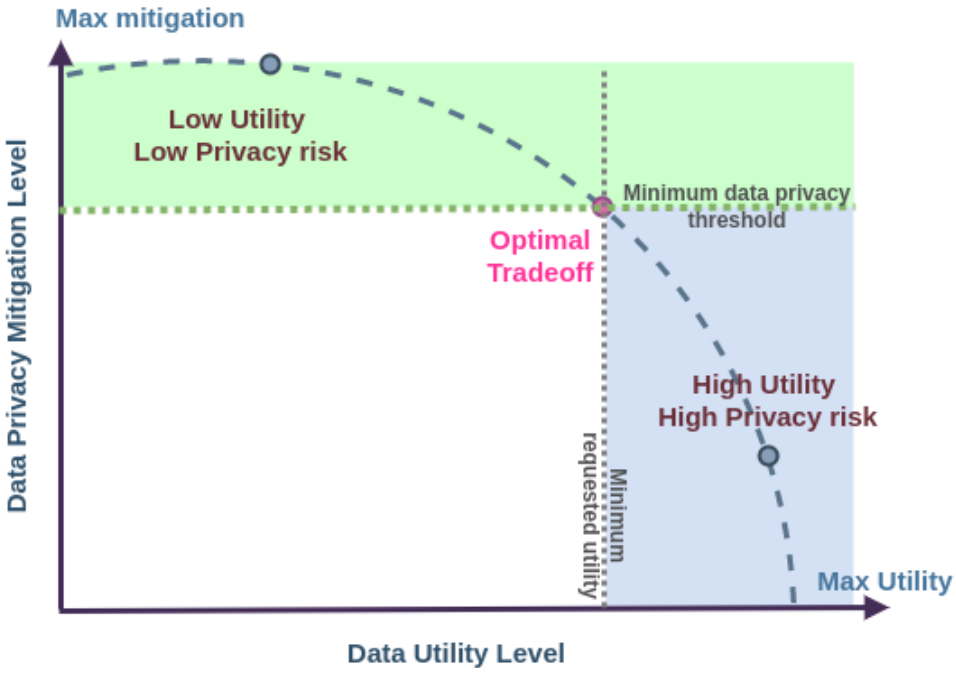


Figure 8.2: A depiction of how privacy-preserving mitigations trade-off utility and sensitivity.

8.6 Data sharing events & Events model

In this section, we introduce our event model, describing the building blocks of behaviour for data exchange systems. This provides a low level of abstraction of the data-sharing events, enabling our analysis of risk within the workflow event model. A single event performs a specific task, such as selecting a data entry, aggregation, and filtering. A workflow model is a composition of events occurring

at possibly physically-distributed sites. Data-sharing events can refer to gaining access to data in storage, transferring data, or processing data with data in execution.

8.6.1 Data in Storage

Data in storage events can be of different types:

1) data read events: reading data from a storage system or source,

2) data write events: writing data to a storage system or destination,

3) data update events: modifying existing data in a storage system, and

4) data delete events: removing or deleting data from a storage system.

Data in storage events can be performed via different methods. Some methods are functionally equivalent but differ in their privacy/security characteristics. A characteristic example is file system access methods. These are fundamental operations for opening, closing, reading, and writing files. These methods are generally considered less secure than their functional equivalents using Object Storage Access or Relational Database Access. This is because file system access methods rely on operating system-level permissions and access controls, which can be vulnerable to unauthorized access if improperly configured. Additionally, file system access may lack encryption mechanisms to protect data at rest.

Data storage events are expressed by specifying an event's type *storage_event*, defining the *data_value*, the data in question, the *storage_location* (ex: database location), and lastly, the method function used *storage_function*. This is expressed as follows:

```
1    storage_event: <data_value, storage_location,
   storage_function>
```

The properties associated with data are discussed in Section 8.5.

8.6.2 Data in Transfer

Data in transfer events can be of different types:

1) data moving events: moving data from one location to another, such as from one storage system to another, or from storage to an execution environment,

2) data import events: bringing external data into the pipeline for processing, and

3) data export events: sending processed data out of the pipeline to another system or destination.

Similarly, transfer events can utilize multiple methods to run, and methods vary in terms of privacy/security, protocols, and implementations. As an example, a requested event can run via a specific transfer function like SFTP (SSH File Transfer Protocol). SFTP is considered one of the most secure options as it provides secure file transfer capabilities over an encrypted SSH connection. It offers authentication, data encryption, and integrity checks. On the other hand, a method function can also run and utilize Remote Mounting, which introduces security and privacy risks depending on the configuration and implementation. It relies on the security mechanisms provided by the underlying protocols (e.g. NFS), which may have vulnerabilities if not properly secured. Transfer events can be expressed as follows:

```
1    transfer_event: <source_location, destination_location,
   data_value, transfer_function>,
```

where the *transfer_event* is replaced with the event type, the *source_location* is the source site data is originating from, similarly, the *destination_location* is where the data will end up, the *data_value* specifies what dataset is the object of this event, and the *transfer_function* is the method this event is utilizing to run.

8.6.3 Data in Execution

Data execution events can be of different types:

1) data processing events: computing (new) output data, possibly using (existing) input data,

2) data transformation events: converting data from one representation (e.g., format) to another,

3) data aggregation events: combining multiple data elements into a single entity or summary,

4) data filtering events: excluding specific data based on certain criteria, e.g., to select only a particular element.

We can run a requested execution event via Secure Multi-Party Computation (SMPC). SMPC is a cryptographic technique that enables secure collaborative data processing among multiple parties. It ensures that sensitive data remains encrypted during computations, enhancing security by minimizing exposure to plain text data. We can also utilize On-Premises Processing with Strict Access Controls methods. By enforcing rigorous access controls, the risk of unauthorized access or data leakage is reduced. On the other hand, utilising General-Purpose Computing Environments (GPCE), such as traditional servers or cloud instances, implicates a lower level of inherent security.

The *compute_type* is defined with the event type, *input_data* and *output_data* define the input and output data of the compute event, respectively, and *processing_function* defines the method function. This is expressed as follows:

```
compute_event: <input_data, output_data, processing_function>
```

A workflow model can be composed with a series of events, and Table 8.1 provides an example workflow model composed of *READ* storage event, *IMPORT* transfer event, and *PROCESS* execute event:

index	workflow tasks
1	$READ\langle EHR_1, loc_a, \text{file system access}\rangle$
2	$IMPORT\langle loc_a, loc_b, EHR_1, FTP\rangle$
3	$PROCESS\langle EHR_1, algorithm(EHR_1), algorithm, GPCE\rangle$

Table 8.1: An example events model

Employing event-based modelling for workflows not only facilitates the assessment of privacy and security risks but also provides a detailed depiction of the services essential for the management and execution of a use case. These use cases align with the format demonstrated in Table 8.1. In addition to this, workflow requests should comprehensively outline data values and specify utility prerequisites imperative for the operationalization of this workflow model. A workflow request has a set of requirements to successfully run with minimal acceptable utility value U_{acc}. To calculate U_{acc} utility's requirement, attributes and weights are set, as formalized in Equation 8.6. Here, users will specify the data type, compute resources, data size, data distance, and data size that should be available to run the workflow and assign 0 and 1 expected values to $E(u_{type})$, $E(u_{compute})$, $E(u_{distance})$, and $E(u_{size})$. Note that, by default, all utility conditions are expected to be met, and hence function E sets all the utility attributes to 1. In addition to α, γ, β, and θ weights reflect the importance of these attributes.

8.7 Privacy Risk assessment

To evaluate the workflow models we defined, privacy and security risk assessment models serve as powerful tools to evaluate and mitigate potential threats to data privacy and security. This is done by systematically identifying vulnerabilities and assessing the associated risks. These models provide a structured framework for quantifying and qualifying risks, allowing for prioritizing mitigations to the most critical areas of concern.

As we previously discussed we make a distinction between security and privacy, and we extend the LINDDIN privacy risk assessment model to assess privacy in particular, which is based on the model published in (116). In this paper, we primarily focus on privacy, but security concerns could align with privacy

concerns and to identify these instances, a number of security models could be used. Namely, the Microsoft approach to threat categorization STRIDE ([8]).

8.7.1 LINDDIN Privacy Model

We address PbD recommendations while running different workflows, and to do that we first calculate the data privacy risk. The privacy risk is defined as the risk of losing control of data usage and disclosure. There are multiple privacy threats that we need to mitigate, as shown in Figure 8.3, and they protect the corresponding features.

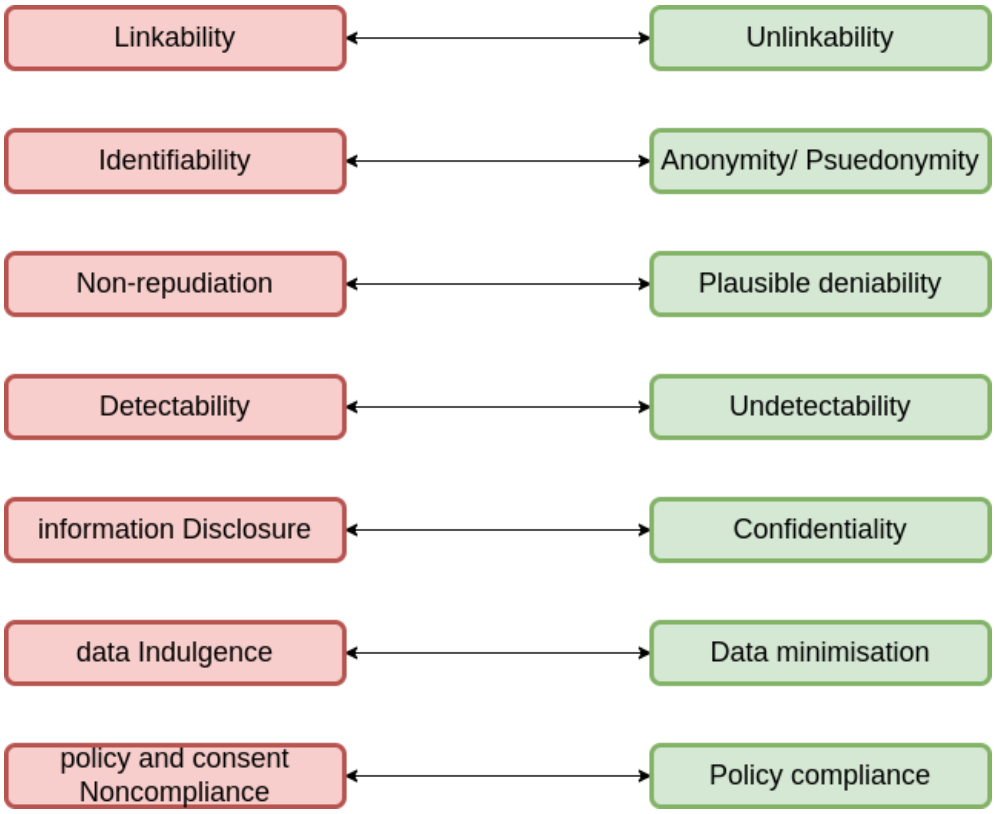


Figure 8.3: The LINDDIN model's privacy features and the associated risks.

Linkability

linkability (L) is the threat that an attacker or unauthorized malicious users can link and relate between two or more data attributes (patient ID, age, condition, etc.). As an example, make a link that patients X and Y are related because they live in the same Postcode and have the same ethnicity. A high degree of linkability can lead to identifiability (and hence unlawful disclosure) and inference that can lead to societal harm. Figure A1 in the **Appendix A** illustrates an example

threat tree where the goal is to exploit linkable data in storage. Figures A8 and A15 illustrate linkability threats relevant to different data stages. We highlight in red texts when privacy threats overlap with security threats modelled in the STRIDE threat categories.

Identifiability

Identifiability (I) is a privacy threat that can result in identifying the subject of a data object, for instance, via aggregating different data sets and interpreting correlations. This is a threat to the anonymity and pseudonymity of patients in research datasets. Figure A2 illustrates the identifiability threats of data with events in the storage stage. Further illustrations can be found in the appendix, relating to data in storage, transfer, and execution events in Figures A.2, A.9, and A.16 respectively.

Non-Repudiation

Non-repudiation (Nr) is a privacy threat when data subjects want to have plausible deniability of events relating to their data. This relates to the threat of failing to retrieve consent for disclosing some data. Similarly, threat trees in the appendix are illustrated in Figures A.3, A.10, and A.17 relating to different data stage events.

Detectability

Detectability (Dt) is a privacy threat where an unauthorised user can determine whether an item exists despite padded data, synthetic data, noise, etc. Note that detectability does not equate to data disclosure, this item is not outright disclosed/known, but its existence can be deduced. Similarly, threat trees are illustrated in Figures A.4, A.11, and A.18 relating to different data stage events.

Information Disclosure

Information Disclosure (iD) is both a privacy and security threat, and it can be quite detrimental to data usage control and privacy. Unlike previous privacy risk categories where data is intentionally but unwisely disclosed, which leads to losing control of the degree of disclosure, the iD threat focuses on unintentional disclosure resulting from weak access control, weak encryption, etc. Example iD threat trees are illustrated in Figures A.5, A.12, and A.19 relating to different data stage events.

Data Indulgence

Data Indulgence (In) is the threat of failing to minimize data during workflow events. This has a direct effect on the linkability and identifiability of data subjects. Example threats are illustrated in Figures A.6, A.13, and A.20.

Policy Non-compliance

Policy Non-Compliance (PNC) is a threat that a workflow event does not adhere to data-sharing policies, and these policies are not (fully) enforced. This could be the result of different vulnerabilities like weak authorisation/authentication of who can write and change a policy, bad policies that do not address edge cases and hence can be exploited, or no or weak enforcement methods. This is illustrated in Figures A.7, A.14, and A.21 relating to different data stages events.

8.7.2 Risk Evaluation

To calculate the privacy risk of a threat, we can use the formula: $risk(threat) = likelihood * impact$. The impact factor is assigned by the system engineers of the consortium. This indicates what privacy features are prioritized, and what feature has little to no concern. Like (119), in this work, we distinguish five levels of impact: critical (1), high (0.75), medium (0.5), low (0.25), and none (0).

The likelihood component considers several factors that affect the probability of a successful privacy threat. One important factor is the *Correlation Level (CL)* of the data with other datasets. Data with low probable correlation has less likelihood of being exploited, while data with medium or high correlation poses a higher likelihood of being targeted.

The *Sensitivity Level (SenL)* of the data is another factor influencing the likelihood. Raw data, which contains personally identifiable information, is more likely to be targeted compared to de-identified or pseudonymised data. Fully anonymized data, where individual identities are impossible to ascertain, has the lowest likelihood of being exploited.

The *Accessibility Level (AL)* determines who can potentially exploit the threat. If the threat can be exploited by consortium members or authorized third parties, the likelihood may be higher compared to threats that can only be carried out by external attackers.

The *Skill Level (SkL)* of the attacker required to exploit the threat is also considered in the likelihood assessment. Attacks that can be executed using simple tools or existing algorithms have a higher likelihood, while those requiring complex tools or multiple steps have a lower likelihood.

The *Dataset Size (DS)* is another influential factor. Larger datasets provide more opportunities for correlations, aggregation, and pattern recognition, increasing the likelihood of a successful privacy attack.

Risk Factors	Correlation Level	Sensitivity Level	Accessibility Level	Skill Level	Dataset Size	Intrusion Detectability
Low Risk	Low probable correlation with other Datasets	Data distance is maximal	Can be exploited by consortium members	Complex (multiple) tools	Small	Easily Detectable with no extra tools
Medium Risk	Medium probable correlation with other Datasets	Data is partially processed to preserve privacy	Can only be exploited by trusted third parties	Existing algorithms/malware	Medium	Can be detected with logging and monitored
High Risk	High probable correlation with other Datasets	Data is raw	Can be accessed by outsiders	Simple tools	Large	Can't be monitored/detected

Table 8.2: The risk likelihood attributes

Lastly, the *Intrusion Detectability (ID)* plays a role in the likelihood assessment. Threats that are easily detectable and can be mitigated through monitoring systems have a lower likelihood, while threats that cannot be easily monitored or detected have a higher likelihood.

By assigning each factor an appropriate value, we can calculate the likelihood component of the privacy risk equation. The values can be set by security engineers subjective to experience, state-of-the-art adversary attacks, and statistical surveys. This comprehensive assessment allows for a more thorough understanding of the potential privacy risks associated with specific threats. Table 8.2 shows these attributes, and rates these values as low, medium, or high-risk values such that $CL, SenL, AL, SkL, DS, ID \in \{0, 0.5, 1\}$, respectively. The likelihood of a single threat t_m is:

$$likelihood(t_m) = \frac{CL + SenL + AL + SkL + DS + ID}{6}, \tag{8.7}$$

such that $t_m \in T_w$, where T_w is the set of privacy risks relevant to running the workflow according to the submitted actions. Then, the workflow risk is

formulated as:

$$r(T_w) = \frac{\sum_{m=1}^{|T_w|} r(t_m)}{|T_w|} \tag{8.8}$$

8.8 The EPI Framework Architecture and Design

After introducing the use cases, the data properties, utility attributes, the event model, and privacy risk categories, we now define the EPI Framework that aims to run a use case adhering to privacy by design. To do that, we delegate different functions into three components: BRANE workflow orchestrator, eFLINT policy reasoner, and BFC orchestrator which handles reprogramming the infrastructure/request.

The EPI components are put in a collaborative architecture to address data-sharing challenges on many levels: application level where we need to run different workflows (potentially distributively) regardless of the heterogenous computing capabilities of the node, policy level where we need to manage and adhere to data-sharing agreements and laws, and reprogramming the overlay network of services to enforce some obligations and rules.

8.8.1 Private & dynamic policies formalisation

First, the institutional policies and laws relevant to the EPI Framework need to be encoded in a way that reasoning about them can take place. As such, they need to be formalised and concretised to the level where they are left unambiguous. After all, the implementation of laws is often left for interpretation to accommodate a particular use case; and in an automated system, this concretisation has to happen beforehand.

There exist multiple languages for formalising these notions. One such language is eFLINT (109), which is designed specifically for reasoning about normative rules. Concretely, it can be used to define a *policy specification* which models and then constraints a particular system. Subsequently, it can be queried about the model's state, and its state can be updated, at which time it will detect if any of the constraints are violated. This makes it a suitable tool for implementing the constraints on a system like the EPI Framework.

8.8.2 Brane: distributed workflow execution

The first two levels of the EPI Framework, the application and policy levels, are implemented by Brane (107), a distributed workflow execution engine designed for use in healthcare (see Section 8.9). In particular, it allows the cooperative

execution of tasks in a workflow on potentially privacy-sensitive data, which is owned by various organisations, in a way that allows its owners to stay in control and exert their policies. This way, it reduces risk using a PbD approach.

8.8.3 Bridging Function Chains model

After the workflow composition with BRANE, and the policy specification, at a lower level, we enforce the conditional policy rules by introducing privacy-as-a-service (PraaS) and security-as-a-service (SecaaS) functions. These functions in the context of the EPI framework are BFs and are often chained into BFCs. The BFC orchestrator adapts the setup to promote privacy and security by creating an overlay network of extra services provisioned by the orchestrator and enforcing mitigations in accordance with the policy and workflow requirements.

Figure 8.4 provides a high-level perspective of the EPI architecture, showcasing its three core components. Once EPI users define workflow events and input data requirements, the execution and orchestration of these workflows take place via BRANE. This module interacts with data-sharing policies, and the BFC model generates setup actions accordingly to provide necessary mitigation. Consequently, the EPI framework finalizes the workflow by integrating the extra BFC services, while concurrently weighing the trade-off between risk and utility associated with this workflow.

8.9 Brane

This section discusses BRANE, the workflow execution engine of the EPI Framework introduced in Section 8.8, in detail so that we can analyse its privacy risks in Section 8.11. BRANE operates in a federated manner. The *orchestrator* is the centralized part that orchestrates the work that decentralised parts cooperatively execute. Each decentralized part is a *domain*, representing an organization. Domains control compute resources and share them, and data, with their peers. In addition, domains are empowered to express constraints on the system by having a *policy reasoner* containing the organisation's data-related policies and determining which actions are *compliant*, i.e., permitted.

The design decisions behind BRANE are motivated by a few core assumptions derived from its intended use case in the medical domain. These are based on previous work that explored the initial design of BRANE (35). The first of these is inherent to organisations in BRANE's use-case:

1. ASSUMPTION. *Organisations often maximise control over their data and minimize their peers' access to their data.*

This is justified by considering the privacy-sensitive nature of the medical data; domains are responsible for keeping that data private. As such, we assert each

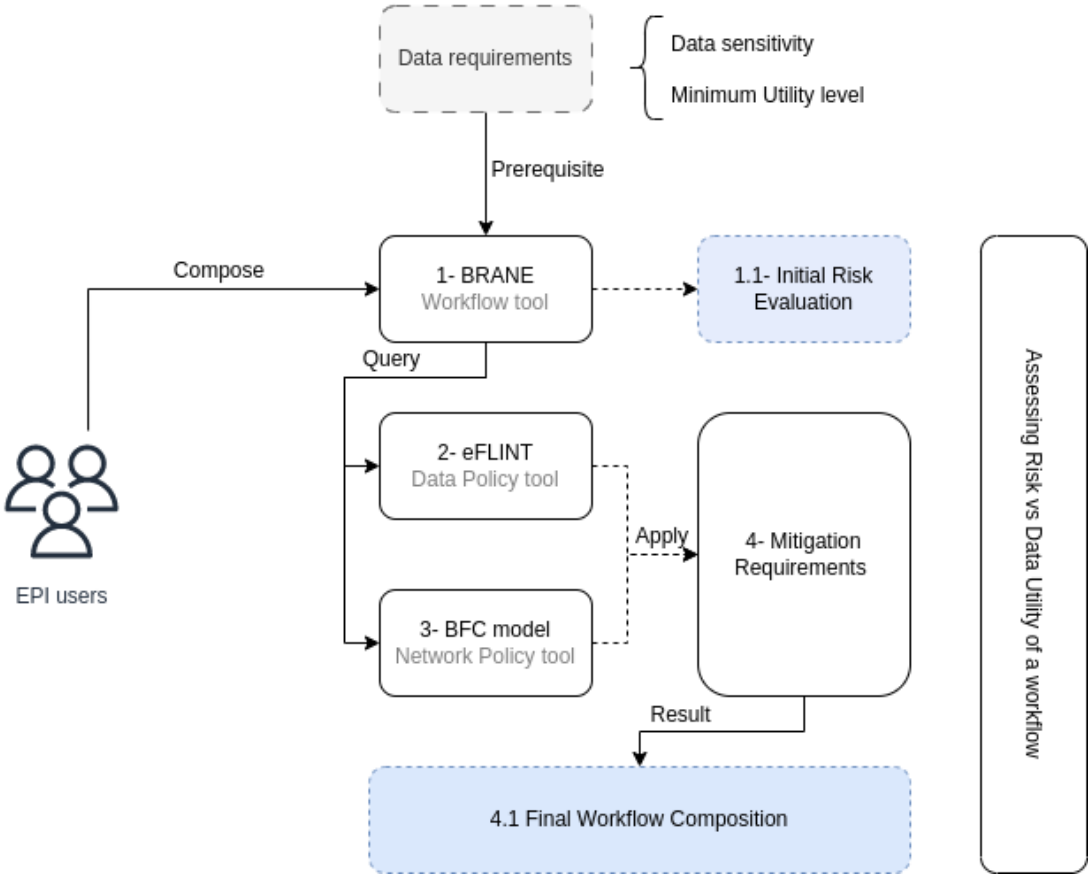


Figure 8.4: A high-level view of the EPI PbD architecture.

domain's control over its actions on its own resources. In turn, domains cannot directly act on the resources of their peers. This fundamental resignation of control over others' actions is an age-old idea, paramount even in the Enchiridion of Epictetus, written in 135 A.D. (e.g., translated in 1955 (34)). Somewhat more recently, it is shared by Mahiru (111), a data exchange and computation system comparable to BRANE.

Assumption 2 follows from domains being autonomous:

2. ASSUMPTION. *Domains cannot be forced by the orchestrator or other domains (not) to act.*

Consequently, BRANE has fundamentally limited control over domains that already access data. For example, BRANE itself cannot compel a domain to delete data at a particular moment[2], and BRANE cannot force a domain to observe the conditions on which it was given access to data. Nevertheless, domains rely on BRANE to orchestrate their cooperation within domains' *constraints*. Services express these constraints in forms convenient to them, e.g., as *policies* encoding access-control rules; (see Section 8.11 for examples).

To simplify matters, we assume that domains control their services such that they act in their own interest, i.e.:

3. ASSUMPTION. *Domain constraints are respected by its own services.*

Constraints expressed in policy reasoners may themselves be privacy-sensitive. For example, as per Article 17.2 of the GDPR (38), "personal data shall, except storage, only be processed with the data subject's consent". The fact that a patient has given consent reveals the presence of their data in the dataset. This is formalized as:

4. ASSUMPTION. *Constraints may be privacy-sensitive.*

Because of this, BRANE does not inspect the state of any policy reasoner directly. Instead, domains react to requests to permit cooperative actions at their discretion, at their own pace, without revealing their underlying rationale.

Finally, for a practical implementation, we assume:

5. ASSUMPTION. *Whether data is present on a domain, as well as data metadata, is non-sensitive information.*

Concretely, the orchestrator may disseminate metadata without consulting policy reasoners. Domains may always control the dissemination of metadata *within* their domains (e.g., to their users), as well as metadata leaving their domains,

[2]Note that domains can always choose to implement some cooperative scheme outside of the framework to provide this control, if necessary.

but this is transparent to Brane. Intuitively, this metadata represents the *public* effects of *private* policies.

Brane's design is discussed in Section 8.9.1. Then, using the assumptions stated above, properties preserved by Brane are defined in Section 8.9.2. The preservation of these properties is elaborated on in Section 8.9.3.

8.9.1 Framework overview

In this section, we introduce the main components of Brane and how they cooperate together on a high level to implement policy-aware workflow execution.

Components

The Brane framework is a collection of six components fulfilling separate functions (Figure 8.5). Three compose the orchestrator and direct the work performed between the domains: the *driver*, which traverses a plan and emits a series of events; the *planner*, which takes a workflow and assigns each of the steps to a domain; and the *global audit log*, which logs the behaviour of the centralised orchestrator and any information shared by local domains to allow ex-post enforcement. Then, every domain consists of the other three components: a *worker*, which takes events emitted by the driver and processes them locally; a *checker*, which encapsulates the domain's policy reasoner; and a *local audit log*, the domain-local counterpart to the global log.

Then there are two domain-local components that Brane interacts with but are typically implemented by third-party services: a backend, which executes tasks as containers; and a data source, which provides access to domain data. These components are the only ones to deal with actual data instead of only control messages. Finally, there is also the user, who interacts with the framework to execute a workflow.

To facilitate domain autonomy (Assumption 2), the implementations of the components can vary between running instances of the framework or between domains within the same instance. It also allows domains to interface with already existing systems, such as patient consent databases. Finally, it affords domains to choose their own level of scalability: *one* running service may provide the functionality of *several* components, or vice-versa.

Interaction between components

The components interact with each other to achieve workflow execution as discussed in (35). In summary, execution begins when a user initiates an interaction with the driver and submits a workflow. Immediately, the driver forwards the workflow to the planner to *plan* it: in this procedure, it compiles the user-friendly,

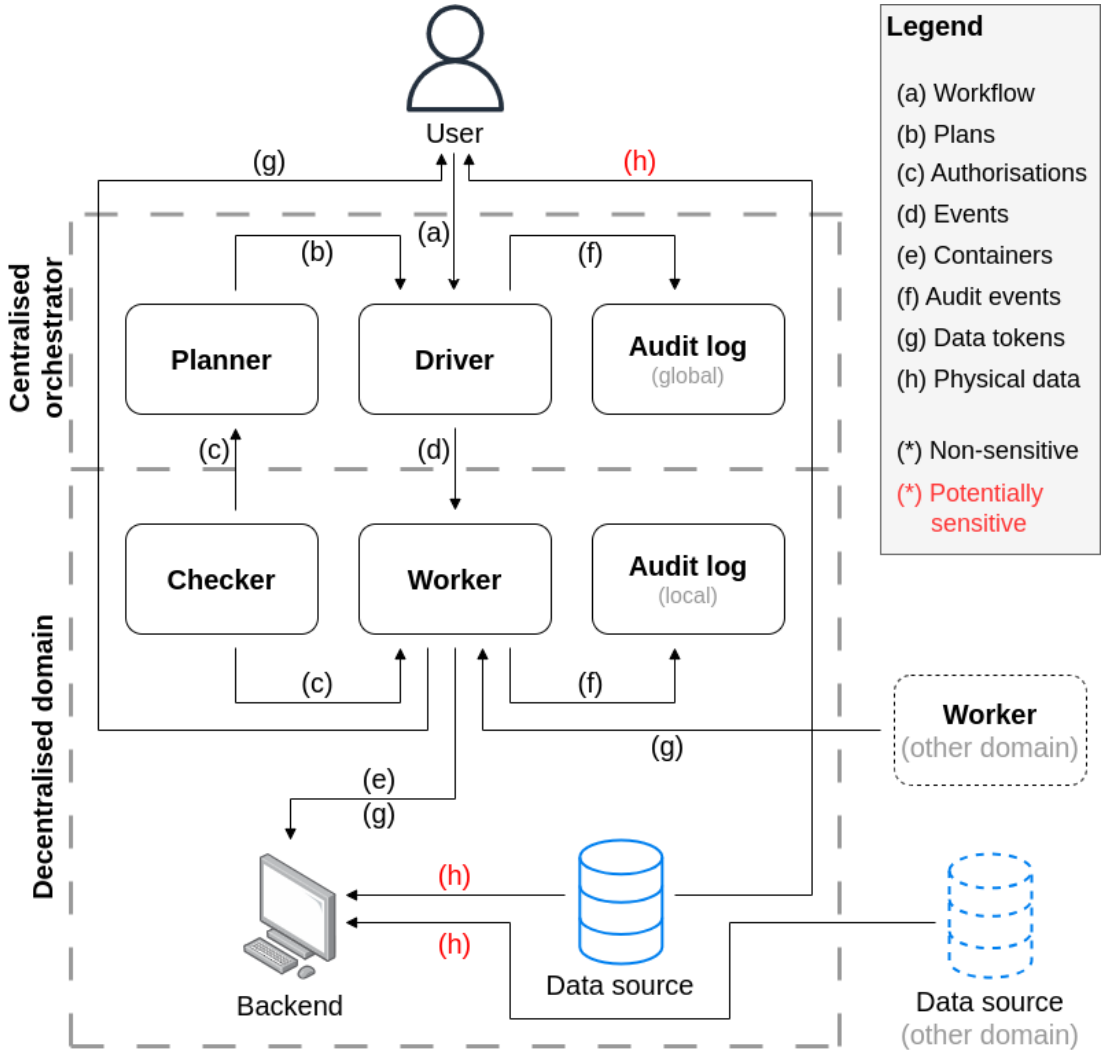


Figure 8.5: Graphical overview of the BRANE framework. The rectangular nodes indicate components. Components comprising the (centralized) orchestrator occur uniquely, while the rest are replicated per domain. Third-party services are represented by nodes with suggestive symbols. Inter-node edges represent communication channels, and are directed as per the "main" information flow; e.g., audit logs mainly receive information.

but abstract, workflow representation to an executable *plan*. Most notably, domains are assigned to tasks and to input datasets, encoded they will execute that task or provide that input, respectively. During this, the planner interacts with checkers to discover which assignments are permitted by the domains' constraints.

Once planned, the planner sends the plan back to the driver, which then starts to traverse it. The driver emits events to workers to execute the tasks encoded in the plan as it does so, including any auxiliary tasks such as transferring data.

The workers process these events only if it is in accordance with the checkers relevant to that event, implementing a secondary check of domain constraints.

When all events have been processed, the driver returns the final result to the user and the interaction ends. Note that this never contains any potentially sensitive data; instead, the user must separately request workers to get access to it, as if they were a worker themselves. This homogenizes all data accesses from the perspective of checkers.

During this process, the audit logs are updated by the other components to keep track of the system's (past) state.

8.9.2 Definition of Safety and Privacy Properties

In this section, we define the fundamental properties of any system using the Brane framework.

Core Ontology

Brane's orchestrator and domains (together, *agents*) communicate essential information in terms of a *core ontology*, the relations in Table 8.3. Concretely, they create, communicate, store, and reason about members of these relations. As such, the properties defined in this subsection are formulated in terms of these relations.

The core ontology formalizes the (compute) events discussed in Section 8.6. To do so, it defines sets of identifiers that refer to Brane entities: *compute* names all compute events that occur in the system; *agent* names all domains; and *data* abstracts over all kinds of data in the system, including implementations of processing functions. In addition, there is a fourth set called *label* that denotes arbitrary strings, although it must at least contain the string Authorized. It acts as a collection of arbitrary metadata assigned by checkers (see Section 8.9.2).

Note that the compute-set only names the events; its parameters are defined using the *input*, *output* and *function* relations. Similarly, compute events are assigned to an agent using the *assigned* relation, and which agents to query before the event may be processed are given by *involves*. Finally, *label*s are assigned to entities using the *labelled*-relation.

In the rest of this section, we denote logical variables using Greek letters or suggestive uppercase letters. For example, A denotes an arbitrary agent. We denote the membership of ϕ in relation Φ as formula $\psi \in \Phi$ or $\Phi(\psi)$, and destructure relation elements into tuples of elements as per Table 8.3. For example, if $output(C, D)$ then D is data.

name	structure	intuition
function	(compute, data)	f in $y \coloneqq f(x)$
input	(compute, data)	x in $y \coloneqq f(x)$
output	(compute, data)	y in $y \coloneqq f(x)$
assigned	(compute, agent)	This worker agent is assigned to process this event.
involves	(compute, agent)	Processing this event requires the authorization of this agent.
labelled	(agent, X, label)	This agent gives this X (data, compute, or agent) this label.

Table 8.3: The *core ontology*: relations with fixed names and structures, but whose elements vary per use case and over time.

Properties Defined via the Core Ontology

The essential properties preserved by the framework are now defined in terms of the Core Ontology.

1. PROPERTY. *Compute events using a data value whose owning agent has never given authorization are not executed.*

This property relies on the assumption that workers always adhere to their own domain's checker (Assumption 3), i.e., workers only compute, access, and transfer data as authorized by their checkers. However, due to Assumption 2, *other* agents will not necessarily follow this restriction. BRANE cannot prevent the data from being propagated further, e.g., via other communication channels, once they have access.

Assumption 4 Assumption 5 reveal a tension checkers must navigate during cooperation. Fundamentally, checkers express constraints on the system which must be obliged. However, checkers may not want to do so when it reveals private information. After all, an attacker may extract that constraint by observing the effects of strategic interactions, as per the algorithm in (105). As such, it becomes important for checkers to be able to decouple their emitted authorisations from their internal constraints, in order to be able to implement countermeasures to such attacks (e.g., introduce acceptable levels of noise). To this end, BRANE generally provides that:

2. PROPERTY. *Checkers fully control their authorizations.*

However, in practice, we expect that many constraints are not privacy-sensitive. Checkers are willing, and incentivized, to share this information, as it affords others making informed decisions. For example, if checkers eagerly authorize plans, plans can be made and executed more efficiently. Generally:

3. PROPERTY. *Agents may share metadata to aid cooperation.*

Agents are free to exchange control information, even beyond the core ontology. For example, they may exchange *hints*, unreliable suggestions of plans considered desirable, serving to guide planning, but not substituting authorization.

4. PROPERTY. *Agents can prove (e.g. to an auditor) that their actions comply with the policies of their peers.*

It is important for any system that requires compliance that it should be possible to detect non-compliant behaviour. In BRANE, this is done by creating a trail of *receipts* which can be used to statically determine if processing an event was allowed by all checkers involved. Moreover, it can also be used to verify whether involvement was computed correctly (see Section 8.9.3).

5. PROPERTY. *Data transfers can be delayed until the required function is being executed, i.e., data is transferred lazily.*

BRANE attempts to maximise data privacy by minimising data transfers; data is only transferred once it must be accessed (see Section 8.9.3). This helps to combat Data Indulgence (Section 8.7.1) and improve performance and compatibility, as it defers access to purpose-built containers.

To conclude, the following properties of how agents should behave can be derived from the previous properties:

6. PROPERTY. *An agent can stay in control of its data if it only enables access to agents for which it has evidence they are assigned to a compute event requiring the data, and that all agents involved with that event have authorised it.*

Effectively, it establishes a norm characterizing well-behaved agents: workers should only access data in order to perform a compute authorized by all involved checkers. By Assumption 2, BRANE cannot guarantee that agents will comply with this norm; so instead, it is up to the individual agents to decide which other agents can be *trusted* to comply.

In the next subsection, background will be provided for how these properties are upheld and what it means for agents to be *involved* or to *authorize* a compute.

8.9.3 Enforcement of Properties

In this section, we explain how some key properties of Section 8.9.2 are preserved in practice.

Centralized Annotation of Computes

The core ontology fixes a set of essential concepts on whose meaning all domains agree; this meaning arises by their special usage by BRANE components. Here, we discuss the relations populated by the centralized services.

Firstly, *input*, *output*, and *function* are populated to reflect the contents of workflows submitted by drivers, also reflecting the formulation of *compute (events)* in Section 8.6.3. Secondly, *assigned* is populated by the planner. Thirdly, *involves* is populated by the orchestrator, relating each compute to domains whose authorization is needed before execution may begin (see Section 8.9.3, to follow).

The centralized control of these relations ensures distinct computes and data are uniquely identified. Other properties make them more meaningful; concretely, **1)** once assigned, each compute's outputs, functions, inputs, and involved agents are fixed, **2)** each assigned compute has exactly one function and exactly one output, and **3)** each assigned compute involves all owners of its input data. These properties let other services reason given incomplete information. For example, a checker observing $(t, x) \in$ *function* can infer $\forall y : y \neq x \rightarrow (t, y) \notin$ *function*.

Decentralized Authorization and Labelling

Checkers communicate asynchronous control information by populating the relation *labelled*. Intuitively, labels establish abstractions over computing events, agents, and data. The domain of labels is unspecified but certainly contains AUTHORIZED, which has a fixed, special meaning, arising from Property 6. Our notion of labels is inspired by *collections* and *categories* in the Mahiru system (111). Thus, preserving Property 2 follows from the preservation of Property 7:

7. PROPERTY. *Agent A uniquely controls which elements of* labelled *match* (A, ϕ_1, ϕ_2)*, i.e., it controls its own labellings.*

This stronger property is enforced by each agent independently. An agent considers $(A, X, L) \in$ *labelled* true only when given proof. Proof is metadata, created and disseminated by agents, for example, via gossip between domains.

We leave the details of our implementation of proof out of the scope of this paper. Here, it suffices to say that our approach centres around agents creating and communicating cryptographically signing *(logical inference) rules* for inferring proof of membership of elements in *labelled*. Crucially, the framework fixes a universal well-formedness criterion for rules: rules inferring $(A, X, L) \in$ *labelled* must be signed by agent A, but may be *applied* by any agent. This approach strikes a desirable compromise. On one hand, this preserves Properties 2 and 7; agents ultimately control their labellings in general and authorizations in particular. On the other hand, rules let agents express authorization in terms of abstract conditions. For example, agent Amy's rule expresses "I authorize any compute whose function is labelled *pseudonymizing* by Bob"; in this example,

Amy effectively delegates the authorization of a particular computing task to Bob.

Note that the usage of cryptographic signatures ensures *non-repudiation* and *integrity* of labellings and authorizations; they can be neither retracted nor forged.

Like the D1LP language ([80]) and Mahiru ([111]), we frame authorization as a *trust management* problem: authorization of compute is proven by attributing it the AUTHORIZED label as the result of reasoning with logical rules. However, our computes are not resources with unique owners responsible for providing authorization. Instead, involvement defines the *set* of agents whose authorizations are needed.

Data tokens

As stated for Property 5, it is practical if data transfers can be performed lazily instead of eagerly. However, from a checker's perspective, it is still valuable to reason about data access as explicit transfers because other domains cannot be relied upon to adhere to access restrictions (Assumption 2).

BRANE solves this tension by representing data access with *access tokens*. Each is a (cryptographically-)signed value encoding where a dataset can be accessed, how it can be accessed, and with which credentials. Workers can exchange these tokens to emulate explicit data transfers and to model which worker has access to which dataset. When a worker processes an event, it can consequently pass the token to a processing function so that it can access only the required data at its own leisure.

The audit logs

The audit logs let BRANE's various components store metadata needed to justify their actions. For example, a worker logs their receipts before beginning work.

The audit log can be seen as a large collection of facts distributed over both the global audit log and the various local audit logs. At runtime, it is unnecessary for any component to have a full overview of all logs, nor do they need to be synchronised to provide a coherent view[3]. To facilitate this, it is fundamentally unordered. Instead, required ordering can be encoded in the facts themselves (e.g., by timestamping).

When an auditor wants to perform an audit, the audit log as a whole must at least provide domains' receipts, information about which events have been started and completed and which checkers existed at the start of each event. BRANE does not require this information to be available automatically; for example, an auditor may have to manually request access to a local audit log of a domain. However,

[3]In fact, this is discouraged, because incoherence between logs likely indicates non-compliant behaviour.

domains are incentivised to ease the task of auditing, as this encourages other domains to trust them with work.

8.10 BFC orchestrator

In this section, we discuss the last layer of the EPI framework, as shown in Figure 8.4. The BFC orchestrators mitigate any threats that were not addressed at the BRANE and policy levels. This PraaS/SecaaS network orchestrator provides the means to provision mitigation methods based on the requested workflow actions, and subsequently the relevant threats. A number of these threats might already be out of concern (low-risk score) due to the current set-up of the environment, for example, if the action submitted is running data transfer with SFTP, then unauthorized access to data is of low risk. While BRANE, powered by policies, guarantees a number of PbD properties, such as data minimisation done by sharing data tokens instead of physical data, it does not ensure minimal risk because it is still highly subjective to the implementation of workflow and the environment setup. The BFC orchestrator deploys preventative measures, and enforces a minimal risk threshold, or otherwise rejects the workflow request. BFC offers a configurable overlay network that is not provisioned at higher layers of the EPI framework, and by that it provides an adaptive infrastructure with privacy risk in mind. It is still essential to mitigate any privacy risk that has not been mitigated by design via BRANE, and to do that there is a need for an adaptive set-up offering PraaS/SecaaS.

In previous work (66)(68), we proposed the BFC orchestrator, a dynamic orchestrator that automates the programming of the underlying networks to secure health data-sharing. Data-sharing is secured by orchestrating and managing services to add security and privacy value to each communicating node, irrespective of each node's capabilities. As mentioned earlier, the BFs are mitigation ready-to-use functions and are implemented with containerisation, which allows flexible instantiation and chaining to enforce increasingly complex policies. In the context of this paper, a communicating node refers to any data flow even within the same domain; management traffic or physical data flow (edges (e), (h), and (g) in Figure 8.5), relating to data in storage, execution, and transfer events.

The *adaptation* of the underlying networks is done after the workflow submission, then the BFC orchestrator evaluates the current setup, taking policy requirements and data requirements as input, to finally translate that into setup actions. For example, data transfer from Domain A to Domain B is only permissible if the privacy risk is below a threshold, which can be done if A can encrypt and decrypt via a stream cypher. The automated setup of the infrastructure is also required to achieve reachability of the end-point nodes, optimal privacy and security across collaborating domains, reasonable network performance, bridging services availability, hardware selection and scalability, and ultimately, abiding

by policy requirements.

The BFC orchestrator is the last layer before actually running the workflow, as part of a use case. The orchestrator operates under Assumption 6 that:

6. Assumption. *There exists an exhaustive threat database.*

7. Assumption. *There exists a many-to-many relation between threats and the BF functions.*

Initially, the evaluation of the environment setup and utilized event function methods is done by cross-referencing the threat database for relevant threats. The relevance of these threats is situational and is directly mapped to the workflow by analysing the pattern formalised via the events model. On top of that, Assumption 7 express a many-to-many relation between said threats and mitigation measures. Meaning that many threats can be mitigated in many ways, by utilizing different BF implementations. The choice of which BF to instantiate, and chain, is dependent on the policy workflow requirements. One thing to note is that these mitigations can affect utility in varying accordance. This is further illustrated in Figure 8.6, such that these measures can be privacy or security mitigations.

Figure 8.6 illustrates the many-to-many relation between threats relevant to a workflow, and makes a distinction between data privacy mitigation (such as anonymisation, aggregation, etc.) and security mitigations (such as network ad-hocs, VPNs, and access control mechanisms).

At this stage, Assumptions 8 and 9 at play, as explained in Section 8.9, the workflow submitted already has checked against the policy. Then the policy will give us an inkling of what is an acceptable risk, and what requirements to enforce. Moreover, as previously explained, and as per Assumption 10, the utility attributes in Section 8.5.2 are set by the user submitting the workflow.

8. Assumption. *Workflows submitted are already checked against policies*

9. Assumption. *Acceptable risk thresholds are set by policy.*

10. Assumption. *The utility attributes are set by the user whilst submitting the workflow request.*

8.10.1 BFC Framework Overview

In this section, we introduce the main components of the BFC framework, the design decisions, and an overview of policy requirements enforcement.

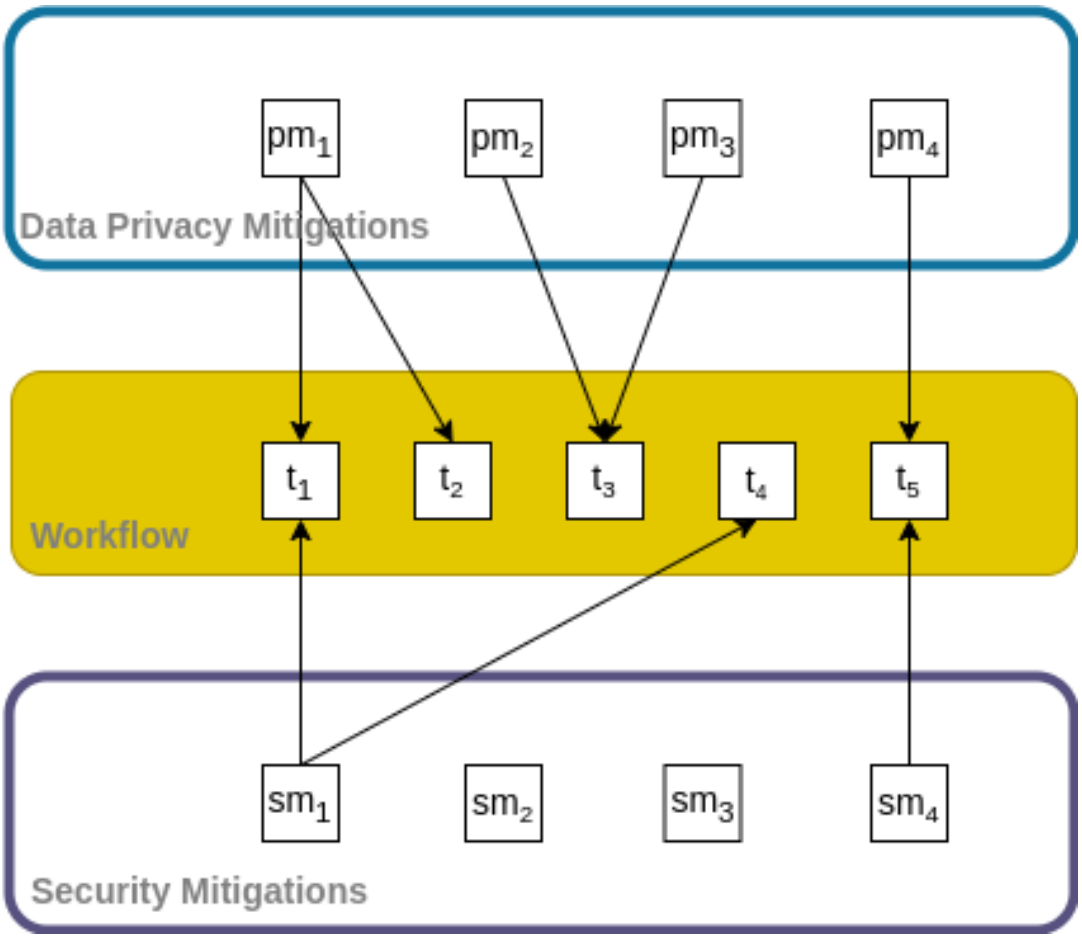


Figure 8.6: Possible data privacy and security mitigations and the relation to threats.

Components

After submitting the workflow request to BRANE, the request goes through the first two layers to plan and ensure compliance. The BFC Framework is deployed to enforce minimal privacy risk by placing extra PraaS and SecaaS; BFCaaS. The framework consists of 4 components spanning over two hierarchical layers control plane and a decentralised plane, as illustrated in Figure 8.7: **1)** BFC orchestrator, which is where service instantiation, placement, and provisioning decisions are made. **2)** Worker nodes, which are Point of Placement (PoP) where BF services are hosted to be instantiated. **3)** Proxy, which holds routing information to enforce dataflow via BF or BFC. 4) BF; containerised instances of these services.

For the implementation of the BFC orchestrator, we need to be able to instantiate on-the-fly services and redirect all actions to go through these services. The implementation is further detailed in previous works (69), where we utilise cilium [4] and deep reinforcement learning to optimally provision and place BFCaaS.

After giving a high-level overview of the BFC framework components and implementation, we will now explain how to reason about the risk and enforce policies.

BFCaaS & Privacy Requirements

To evaluate the current setup, we need to refer back to, Assumption 6 and consider, a set of possible privacy threats $T_{workflow}$ that are relevant to running the workflow. BF is the set of possible mitigation functions, such that $bf_j \in BF$, and BF is:

$$BF = \{sm_k | k = 1, ..., d\} \cup \{pm_l | l = 1, ..., v\}, \tag{8.9}$$

sm_k and pm_l are members of the data privacy and security measures sets, respectively. Each workflow in accordance with the requested actions has a set of expected known threats, and these threats are more or less relevant ($risk \approx 0$) according to the set-up in place relating to the type of function utilised to run a transfer, storage, or execution events.

As per Assumption 7, a single threat can be mitigated via a single bf_i or chain of bf; $BFC_j \in BFC$, such that:

$$BFC_j \in \mathcal{P}(BF), \tag{8.10}$$

A chain is a set of bf combinations, where you have one or more bf, and $\mathcal{P}(BF)$ is the power set of BF. Moreover, the Property 8 showcases the degree of mitigation these measures have relating to a threat.

[4] https://cilium.io/use-cases/service-mesh/

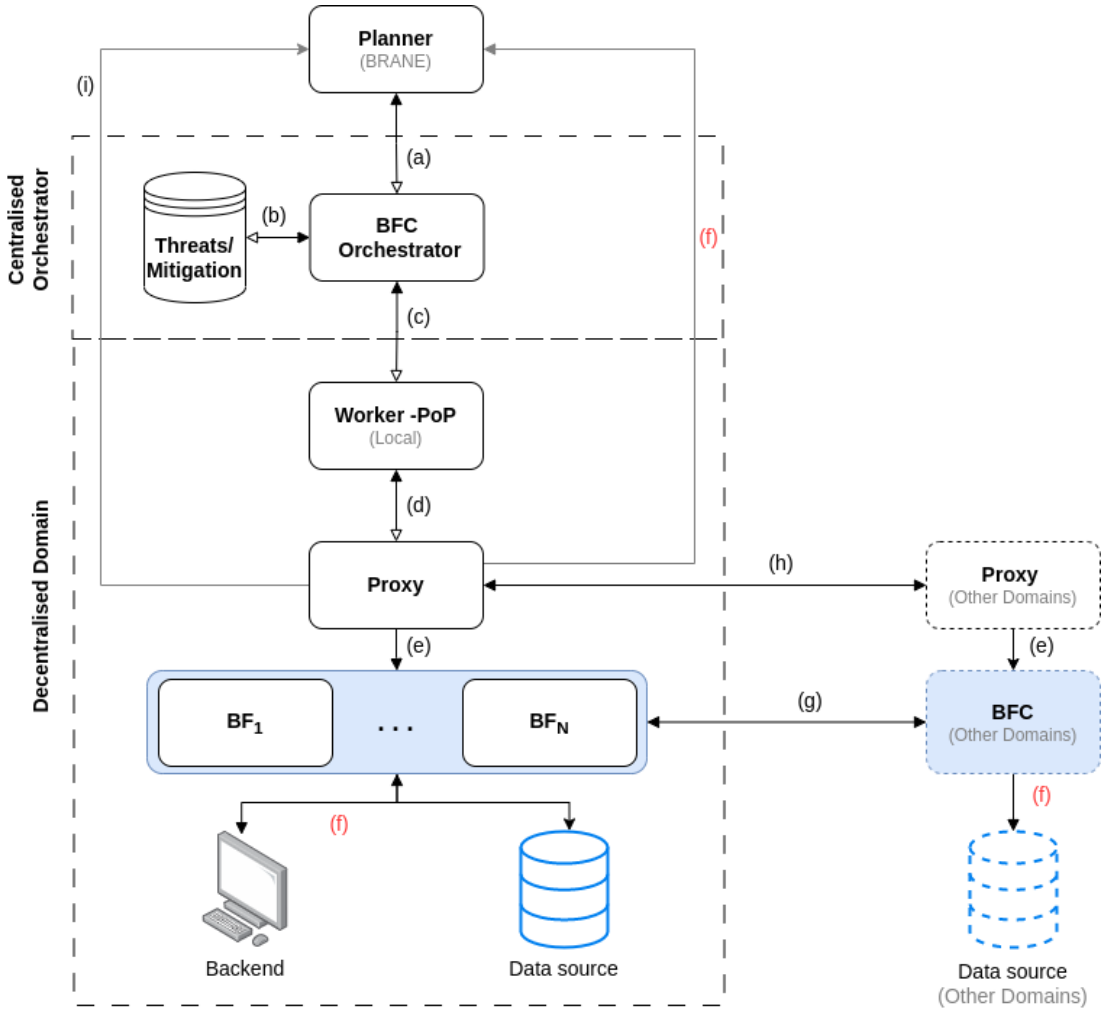


Figure 8.7: Overview of the BFC Framework. There are two main layers to deploy the BFC framework, the centralised orchestration layer (the central place) and the decentralised domain(s) (services in this layer are duplicated / domain in the system). The edges illustrated between services are: (a) Event plans, (b) threat queries, (c) Service management, (d) route configuration, (e) rerouted dataflow, (f) Physical data, (g) distributed service chaining, (h) domain-to-domain dataflow, and (i) Data tokens.

8. PROPERTY. *A bridging function or a chain of bridging functions can mitigate a threat to a varying degree.*

11. ASSUMPTION. *A bridging function or a chain of bridging functions are implemented securely, hence added inherent risk is negligible.*

In relation to Property 8, we measure the mitigation ratio by defining the mitigation factor f, such that $t_m \in T_w$:

$$f(t_m, bf_i) = \begin{cases} 1, & \text{fully mitigated} \\ \frac{\delta r(t_m)}{r_0(t_m)}, & \text{otherwise} \end{cases} \tag{8.11}$$

where $\delta r(t_m) = r_0(t_m) - r_1(t_m)$, such that r_0 and r_1 are the risk of threat t_m before (initial) and after applying the mitigation measure bf_j. $f(t_m, bf_i) \leq 1$ and $f(t_m, bf_i) > 0$ because if $r_0(t_m) \approx 0$ then $t_m \notin T_w$. $\delta r(t_m) \geq 0$ building on Assumption 11, where BF functions implemented by security experts. Moreover, to calculate the cumulative effect of applying the chain BFC to mitigate threat t_m:

$$f(t_m, BFC_j) = \begin{cases} \sum_{p=1}^{|BFC_j|} f(t_m, bf_p), & \text{If} < 1 \\ 1, & \text{otherwise} \end{cases} \tag{8.12}$$

The conditional function ensures that the mitigation factor $f \leq 1$, meaning it is already fully mitigated if $f = 1$. Moreover, the remaining risk of a threat after applying BFCaaS is:

$$r_r(t_m) = r_0(t_m).(1 - f(t_m, BFC_j)), \tag{8.13}$$

where $0 \leq r(t_m)_r \leq 1$.

Moreover, to calculate the remaining risk of the total workflow $r_r(T_w)$, then we look at all the risks of all the threats, and the different mitigations applied. The remaining risk of a workflow is calculated such that:

$$r_r(T_w) = \frac{\sum_{m=1}^{|T_w|} \left[r_0(t_m) \cdot (1 - \sum_{j=1}^{j=N} f(t_m, BFC_j)) \right]}{T_w}, \tag{8.14}$$

where N is the total number of BFCaaS mitigation deployed.

8.10.2 Privacy & Security vs Utility

Other than the mitigation factor, utility requirements also factor into the decision of BFCaaS provisioning. Some BFs might affect the utility depending on the type of function, so as an example anonymization function can increase the sensitivity,

hence decreasing utility. This effect is formalized as utility factor g, and $g : U_{bf_{(i-1)}} \times bf_i \rightarrow U_{bf_i}$, such that U_{bf_i} is the new utility of $U_{bf_{(i-1)}}$ after applying bf_i. If $U_{bf_i} = U_0$; the initial utility, then bf_i has no effect on utility.

The resulting utility after applying the BFCaaS is formalised as U_r, such that:

$$U_r = g(U_0, BFC) = g(U_{BFC_{j-1}}, BFC_j) | 1 < j < N, \tag{8.15}$$

in Equation (8.15) U_r the calculated in terms of initial utility U_0, and the set of all mitigation chains BFC.

$$g(U_0, \phi) = U_0, \tag{8.16}$$

where in Equation (8.16) no mitigations were applied, hence $BFC = \phi$, and no effect on utility occurred.

9. Property. *BFC mitigation decision is a constrained optimizing problem.*

10. Property. *BFC are deployed to minimize privacy risk with utility in mind.*

11. Property. *A workflow request can fail if utility or privacy requirements are not met.*

With that in mind, searching for the optimal BFC set of mitigation chains is dependent on the utility and privacy risks acceptable U_{acc} and r_{acc}, as discussed in Section 8.5.2 and Section 8.7.2. This search problem is a constrained optimising problem, in reference to Property 9, where the BFC orchestrator aims to minimise privacy risk and maximise utility according to the following constraints:

$$U_r \geq U_{acc} \tag{8.17}$$

$$r_r(T_w) \leq r_{acc} \tag{8.18}$$

This problem is addressed with a heuristic search algorithm, as in Algorithm 6. The problem is simplified to find the best effort BFC mitigation chains with Property 11 in mind. We start by assigning BFC, U_r, and $r_r(T_w)$ to initial values in lines 1-3. We set λ to 1 in line 4, to prioritize minimizing privacy risk (privacy risk has maximum priority in the first iteration of the search). Loop over all threats in T_w, and try to find the best mitigation (chain) for this threat by finding the set combination $BFC_j \in \mathcal{P}(BF)$ that minimizes risk and maximizes utility in line 8-9. Then check the resulting effect of this BFC_j in lines 11-12, and compare it against the acceptable values. If this fails, then remove BFC_j from the BFC sets that will be deployed, and tune the λ parameter to change the search weight and prioritize the utility more in lines 18-19. Try again until $\lambda < 0$, then exit the while loop, and look into other threats. After going over all threats and adding mitigation (chain) sets to BFC, check the effect of the whole set as in Equations (8.14) and (8.15), and reject or accept the workflow according to the constraints.

Algorithm 6 Search for Optimal Mitigations

```
1: procedure MITIGATE(T_w, BF, r_acc, U_acc)
2:     BFC ← φ
3:     U_r = U_0
4:     r_r(T_w) = r_0(T_w)
5:     λ = 1
6:     ordered list T_w
7:     for all t_m ∈ T_w do
8:         while λ ≥ 0 do
9:             BFC_j ∈ P(BF)
10:            BFC_j = argmin_{BFC_j}(λ · f(t_m, BFC_j) − (1 − λ) · g(BFC_j))
11:            BFC ← BFC_j
12:            r_r(T_w) ← new r_r(T_w)
13:            U_r ← g(U_0, BFC)
14:            if r_r(T_w) ≤ r_acc AND U_1 > U_acc then
15:                Success!
16:            else if r_r(T_w) > r_acc AND U_r > U_acc then
17:                Continue to the next threat
18:            else
19:                BFC − {BFC_j}
20:                λ ← λ − 0.1
21:            end if
22:        end while
23:    end for
24:    if r_r(T_w) > r_acc OR U_r < U_acc then
25:        Reject workflow
26:    else
27:        Return BFC, r_r(T_w), U_r
28:    end if
```

8.11 DIPG USE CASE: A walk-through

Previous sections laid out all the use case concepts, the risk assessment method, and the EPI Framework components. This section expands on the first use case; the DIPG; and provides a walk-through that applies all said concepts to showcase the EPI framework's capabilities in privacy risk minimization. Since not all the details of this use case are known, we make assumptions to concretely reason about it, and give example setups of the data exchange environment and the associated EPI framework decisions.

8.11.1 Data properties & Event model

Table 8.4 specifies an example DIPG workflow request, where an EPI user with a data scientist role wants to read and transport a small DIPG dataset EHR_1 from the DIPG repository location loc_{DIPG} to the hospital as a destination loc_{hosp}.

The data properties related to this EHR_1 record are sensitivity and utility. First, in this use case, the data maintained in the DIPG repositories is pseudo-anonymised, hence the data sensitivity is medium and therefore can be set to 0.5 ($distance = 0.5$); thus is not completely anonymized.

Second, the required data utility attributes for this workflow are set by the user. The requested data type for this workflow is EHR, where EHR_1 is expected to be made available. No compute power is needed (no processing required whilst executing this workflow), meaning $r_{req} = 0$. Moreover, a higher distance is acceptable for this workflow (> 0.5), and the distance required can be $distance_{req} = 1$. Subsequently, the EPI framework is allowed to further anonymise the data without compromising the workflow.

With $E(u_{type}) = E(u_{compute}) = E(u_{distance}) = E(u_{size}) = 1$, further assume that type, distance, and size requirements are equally important, and compute resources requirement is not. Hence, the utility parameters are set to $\alpha = \beta = \theta = \frac{1}{4}$. As a result, acceptable utility is a weighted sum of the expected attribute values $U_{acc} = \frac{3}{4}$.

The workflow events are modelled in Table 8.4, where there are two events: storage event of type read with SQL as the read function method, and transfer event of type move with FTP as the transfer function method.

index	DIPG workflow task
1	$READ\langle EHR_1, loc_{DIPG}, SQL\rangle$
2	$MOVE\langle loc_{DIPG}, loc_{hosp}, EHR_1, FTP\rangle$

Table 8.4: The DIPG event model

8.11.2 Initial Risk

According to the utilised function methods specified in the events model, the workflow might be vulnerable to several threats. Table 8.5 showcases an example threats list relevant to the DIPG workflow. In the table, we also set the risk attributes according to Section 8.7.2. Relevant threats are primarily related to the data in storage and data in transit stages events. These threats concern one or many privacy risk categories.

To calculate the initial risk, we go over the 14 threats' risk attributes, and average the total risk as in Equation (8.14). In this example, the initial risk is:

$$\begin{aligned} r(T_w) &= \frac{\sum_{m=1}^{14} r(t_m)}{14} \\ &= \frac{\sum_{m=1}^{14} likelihood(t_m) \cdot impact(t_m)}{14} \end{aligned} \tag{8.19}$$

The exact answer is subjective to the set impact for each threat, and this is set at a higher consortium level. For this example, assume that the impact of these threats is the maximum value ($impact = 1$), hence $risk = likelihood$ and the initial risk is then only dependent on the set risk attributes, such that:

$$r(T_w) \approx 0.54 \tag{8.20}$$

Threat Name	Stage	Category	Correlation	Sensitivity	Accessibility	Skill	Size	Detectability
Unauthorised Access	Storage	iD	M →L	M → L	H →L	H →M	L	H →M
Linkable Data Attributes		L	M →L	M → L	L →L	L	L	H →M
Identifiable Data Access		I	M →L	M → L	L	L	L	H →M
Profiling and Tracking		Nr	M →L	M → L	H	M	L	H →M
Long Retention		Nr	M →L	M → L	M	H →M	L	H →M
Unlawful Retention		Nr	M →L	M → L	H	H →M	L	H →M
Policy Modification	Transit	Nc	M →L	M → L	M	H →M	L	M
Eavesdropping/Sniffing		iD	M →L	M → L	H →L	H →L	L	H
MitM		iD	M →L	M → L	H →L	H →L	L	M
Traffic Analysis		L	M →L	M → L	H →L	L	L	H
IP Tracking		L, I	M →L	M → L	H →L	L	L	H
Metadata Leakage		iD, Dt	M →L	M → L	H →L	H	L	H
Oversharing of Data		Nc, In	M →L	M → L	M →L	H →M	L	M
Unlawful Sharing		Nc	M →L	M → L	M →L	H →M	L	M

Table 8.5: The DIPG workflow's relevant threats and the example risk attributes. After applying the EPI Framework, some threats are mitigated, translating to lower risk estimations. The new values are colour-coded, where grey values are mitigated by Brane 's design, orange values are mitigated by the chosen eFLINT policies, and the blue values are mitigated by the BFC orchestrator.

8.11.3 The EPI framework's Mitigations & Residual Risk

This section focuses on demonstrating how certain privacy risks can be mitigated with the EPI Framework. Starting with BRANE PbD mitigations, then the formalization of policies stipulated in privacy regulations and contracts, and lastly BFCaaS setups.

BRANE mitigations

Because of its Privacy-by-Design principles, Brane allows for the mitigation of certain risks. Mostly, this is done by empowering the policy reasoners on a domain to dynamically apply the mitigations. For example, Brane has the option to lower the risk of an attacker getting access to Linkable Data Attributes, but only if the policy reasoners implement the correct constraints on Brane. As such, most of these mitigations are discussed as particular policies in the next section.

However, there are a few mitigations that Brane always provides regardless of policy. Mainly, by making use of its audit logs, Brane improves the detectability of certain threats. In particular, any threat relating to data access for which the involved checkers have to give authorization can be detected as a discrepancy in the logs' proof trees. However, the attacks can only be detected if the violation occurs within the framework itself: for example, it can be detected if a domain processes a workflow on a dataset to which it should not have access, but not if a domain is hacked and the data is copied manually without leaving a trace. As such, the detectability risks of such threats are reduced to medium risk (M).

Further, the skill level required to perform attacks is lowered in a lot of cases by the fact that Brane only needs very limited access to a domain through well-defined interfaces. This makes it harder for attackers to perform Policy Modification on a particular domain, which further reduces the skill risk of other threads. However, because the policies are residing on domain infrastructure and not Brane-managed infrastructure, the skill required to attack the policy is also dependent on how well-secured the domain is.

eFLINT mitigation

In this section, a brief explanation of policies that potentially mitigate some of the risks listed in Table 8.5 is provided. All examples in eFLINT are shortened for brevity, a detailed explanation of the code can be found in previous work ([108]).

Granting access to authorized users only: An access control mechanism is one way to mitigate linkability. Access control allows organizations to restrict access to data values, for example, based on the sensitivity of data and/ or based on the roles of users. For the DIPG use case, access to the data values is granted only to users who are affiliated with a member organization and have a project

proposal approved by SIOPE DIPG/DMG. The following eFLINT fragment formalizes that:

```
Fact approved-project Identified by project * member.
Extend Act read Holds when (Exists project, member:
  selected(asset,project) && approved(project,member)
  && affiliated-with(user,member)).
?Enabled(read(<user>,DCOG,<EHR1>)).
```

Listing 8.4: Grant access to researcher with an approved project

Listing 8.4 specifies that the action "read" is to be understood as accessing a file, and it is enabled when there exists a data value(asset) that is selected for an approved project and the user that is requesting access is affiliated with a member organization. The query is evaluated to be true if a user is affiliated with a member organization and the user has a project approved for which EHR1 has been selected. Such a policy ensures that unauthorized access does not take place and reduces the high risk associated with unauthorized access, in Table 8.5, to a minimum.

Granting access to authorized purposes: The GDPR dictates that data collected for a certain purpose be used only for said purpose unless a different legal basis applies. In addition to authorizing users based on approved projects and affiliations, access is granted based on authorized purposes specified by data subjects or controllers. By doing so, the probability of unlawful data sharing is minimized. Furthermore, restricting access based on authorized purposes restricts organizations from re-purposing data values, since controllers can't re-purpose data values unless authorized by data subjects or other legal basis applies.

The policy in Listing 8.5 specifies that the action of collecting personal data is affiliated with a legit purpose and the user is part of an authorised project.

```
Fact accurate-for-purpose Identified by data * purpose.
Extend Act read Holds when (Exists project, member:
  selected(asset,project) && approved(project,member)
  && affiliated-with(user,member))
  && accurate-for-purpose()).
?Enabled(read(<user>,DIPG-purpose,<EHR1>)).
```

Listing 8.5: Grant access to researcher with an approved purpose

Finally, restricting access to data values for a specific purpose minimizes linkability by restricting access to the amount of contextual information one can gather from certain data values, making it harder to link data values to a specific individual.

8.11.4 BFCaaS

The last step of mitigation setup is to go through the BFC orchestrator and instantiate BFCaaS in compliance with r_{acc} set by the policy, and u_{acc} submitted by the EPI user. At this stage, $r_r(T_w)$ is calculated again, such that:

$$r_r(T_w) \approx 0.18, \tag{8.21}$$

where this indicates the remaining risk after deploying BRANE and formalizing the policy with eFlint.

To further minimize this risk probability, the orchestrator deploys two example services: anonymization and secure file transfer protocol $\in pm$ and $\in sm$, respectively. With these services, risk attributes like sensitivity decrease to L, and subsequently, this decreases correlation as well where the anonymization function replaces correlatable data values with randomized new values. This is also compliant with the acceptable requested utility. Moreover, the secure transfer protocol decreases the probability of accessibility relating to packet sniffing, MitM, and similar attacks. As a consequence of the traffic being encrypted over the network, it takes more complex tools to exploit (some skill attributes are lowered to L).

8.12 conclusion

Medical data sharing is essential for research advancement, but Privacy by Design (PbD) principles are as important as ever, especially in dealing with medical data. The EPI Framework facilitates data sharing while ensuring compliance with PbD principles. This framework effectively manages privacy risks, maintains control over data, and governs disclosure. It is constructed with the utilization of BRANE, a workflow execution engine, and the BFC orchestrator, a BFC-as-a-Service network orchestrator. We introduced a privacy risk model to quantify the privacy risk associated with data sharing. We evaluated the mitigated risk in reference to data utility and the predefined acceptable risk as per relevant policies.

Chapter 9

Conclusion

Data sharing has become fundamental to advancing medical research, facilitating breakthroughs in understanding, preventing, and treating health conditions. The evolution towards personalized medicine represents a paradigm shift, emphasizing tailored treatments based on individual patient characteristics. This approach acknowledges the diverse influences of genetics, environment, and lifestyle on health outcomes, advocating for precise interventions tailored to each patient's unique profile. As larger datasets become available, the promise of personalized medicine grows, fuelled by machine learning models and advanced data analytics.

The concept of DHT further amplifies the potential of personalized medicine by creating digital models that mirror individual physiological systems or healthcare institutions. By leveraging diverse patient data and cutting-edge technologies, DHTs pave the way for innovative healthcare applications, ranging from drug development to treatment optimization.

However, the realization of personalized medicine hinges on robust data-sharing mechanisms and collaborative efforts across healthcare domains. Challenges surrounding data privacy, security, and interoperability underscore the need for dynamic infrastructures capable of adapting to evolving healthcare infrastructure landscapes. Developments like the EPI framework offer a glimpse into the future of healthcare infrastructure, integrating policy-driven data sharing with advanced networking technologies.

The EPI framework; as discussed in this thesis; considers the trade-offs between security, privacy, and network performance as a critical design prerequisite. Enhancing security often requires additional resources, such as encryption and multi-factor authentication, which can introduce network overhead. Similarly, robust privacy measures, including data anonymization and access controls, can introduce latency, complexity, and lower data utility. On the other hand, prioritizing performance might necessitate compromises in security and privacy, leaving systems vulnerable to breaches, unauthorized access, and non-compliant data-sharing. As we demonstrated, managing this trade-off within the EPI frame-

work is mapped to the use cases' policies, acceptable security and privacy risks, and performance requirements. This is done to optimize all aspects to meet the specific needs and constraints of a health data-sharing use case.

The presented thesis highlights the key contributions in this discourse from the design of data-sharing frameworks to the implementation of adaptive infrastructure. This thesis proposes the EPI framework to promote more secure and reliable health data sharing in terms of network performance and resource utilisation.

The main contributions of this work are:

- A conceptual policy-adaptive framework for health data sharing, and we implement the proof of concept with Kubernetes to process distributed workflows.
- Autonomous policy reasoning and infrastructure setup according to a requested workflow. The reasoning is done by aggregating the feasible data flows according to the infrastructural feasibility and the allowed data flow, in compliance with the policy.
- The implementation of the EPI proxy that redirects, manages, and chains traffic to enforce data-sharing policies.
- An AI-powered provisioning tool to effectively place EPI functions and assign computing resources.
- A *privacy-by-design* data-sharing framework with risk minimisation approach, and in our methodology we also consider relevant privacy threats, data utility, and mitigation techniques.

Due to the listed contributions of tool design, and development and deployment of the proof of concept, we were able to answer the posed questions. Starting with the subquestions:

- RQ1: "How can we address open policy, security, and computing data sharing challenges via building a dynamic infrastructure framework to deploy DHT use cases?"

The proposed EPI framework in this study comprises three primary components: 1) the policy reasoner, 2) the infrastructure orchestrator, and 3) the workflow orchestrator. These components work collaboratively to manage policy, security, and computing aspects, addressing the specified concerns. Chapter 3 provided an in-depth exploration of this solution by presenting the architecture and design of the EPI framework. The workflow initiation, facilitated by BRANE, undergoes a formalization process and is then assessed against the defined policies to ascertain compliance. These policies encompass both data-level and network-level considerations. In Chapter 4, we explained the logical model guiding the aggregation

and reasoning processes for different policies. This comprehensive approach results in the implementation of policy-compliant workflow orchestrations through the adoption of the EPI Framework, effectively answering the question at hand.

We developed a robust logic model, enabling the policy orchestrator to process and scale with $O(n^2)$ CPU time, even as the complexity of heterogeneous infrastructural areas increases across data-collaborating nodes. We conclude that the EPI framework is efficient in addressing multifaceted challenges by integrating policy reasoning with infrastructure and workflow orchestration. The policy reasoner ensures that all processes adhere to predefined security and privacy standards, mitigating risks and enhancing compliance. The infrastructure orchestrator dynamically manages resources to align with policy requirements, optimizing performance. Meanwhile, the workflow orchestrator coordinates tasks efficiently, ensuring the smooth execution of operations within the set policy constraints. This integrated approach fortifies the system's resilience against potential threats. This study offers a valuable model for future implementations in various contexts requiring stringent policy adherence and resource management.

- RQ2: "What are the performance tradeoffs when employing different packets' redirection methods and bridging functions to enforce network policy-compliant routes under different workloads?"

To answer this question we implemented the EPI proxy using NGINX and SOCKS tools. Additionally, we implemented some *Bridging Function Chains* that could be deployed within the EPI framework setup like firewalls, encryption, and decryption functions. We evaluated and analysed the tradeoffs in a Kubernetes environment and simulated multi-domain setups. In Chapter 5, our research showed that when comparing the different proxies, there exists a tradeoff in terms of port scalability, optimisation, reconfigurability, dynamicity, and security. Moreover, the SOCKS-based proxy outperforms the NGINX-based proxy in terms of processing rate and it supports all traffic types. On the other hand, the NGINX-based proxy has a lower overhead. In Chapter 6, we demonstrated a correlation between the BFC topologies and the reductions in received packet rates, indicating that there is a computation, latency, and processing rate tradeoffs with different BFC topologies.

In conclusion, the SOCKS-based proxy's superior processing rate and compatibility with all traffic types make it a versatile choice for environments requiring high throughput and broad traffic handling capabilities. However, this comes at the cost of higher overhead compared to the NGINX-based proxy, which, while offering lower overhead with ¡1ms running 120 consecutive requests and thus potentially better performance in specific scenarios, lacks the same level of processing efficiency and versatility. Additionally, the implementation and evaluation of BFCs revealed significant impacts on packet rates and processing latencies, highlighting the importance of carefully selecting and configuring BFC topologies

to balance security and network performance. The evaluation conducted in this study provides valuable insights into the optimal deployment strategies within the EPI framework, guiding future implementations towards achieving a secure and effective network infrastructure.

- RQ3: "How can we automate an adaptive Service Function Chain Provisioning on available network Points of Placement candidates to run a data-sharing request under different use cases' requirements?"

To answer this question we first formalised a comprehensive problem statement of SFC provisioning. We formalised this problem as an optimal restricted search problem, such that the search restrictions are the different use case requirements. In Chapter 7, we implemented and deployed heuristic and AI-based search algorithms to optimally place and chain SFC according to the requested policy, and under the use case requirements restrictions. Our research shows that heuristic-based approaches are adequate when latency overhead is tolerable according to the use case requirements, offering a practical solution for optimizing performance under static operational conditions. However, for scenarios where minimizing latency is paramount, HDQL tools record the lowest latency (halved compared to traditional heurisitic-based methods) and highest CPU utilization rate ($\approx$ 0% un-utilised allocated CPU). Moreover, HDQL is robust against fluctuating network traffic and stringent latency requirements. Deploying HDQL within the EPI framework ensures that network policy-compliant routes are robustly provisioned to meet different use cases' requirements.

- RQ4: "How to orchestrate the EPI framework services according to privacy by design principles by comprehensively modelling privacy probabilities and mitigating risks of DHT data-sharing workflows according to privacy-defined attributes?"

In Chapter 8, we explored the performance of the EPI framework according to the PbD considerations. We proposed the privacy risk assessment model to quantify risk and evaluate our framework. The workflow submitted to the EPI framework specifies 1) Data properties the workflow requires to have access to, 2) and data sharing events within the workflows. The LINDINN privacy model defines privacy as the threats of the Linkability of data, Identifiability of data, Non-repudiation of data, detectability of data, information Disclosure, data Indulgence, and policy and consent Noncompliance. The EPI framework takes into consideration these privacy risks by design.

The EPI framework is proven to mitigate privacy risks when running use case workflows. Privacy risk is quantified by assessing the risk of probability and the impact of a successful exploit. Notably, the EPI framework decreased the privacy risk of running the DIPG workflow from 0.54 to 0.18, demonstrating significant improvements in safeguarding sensitive data.

These improvements underscore the framework's ability to enhance policy compliance, privacy, and security. By incorporating a robust privacy risk assessment model, the EPI framework addresses critical concerns of data linkability, identifiability, and non-repudiation, among others. This proactive approach to privacy by design ensures that data-sharing workflows adhere to stringent privacy standards, thereby reducing the likelihood and impact of data breaches.

Now that we have answered the subquestions we can answer the main research question:

RQ: How can we effectively support health data sharing use cases for heterogeneous parties collaborating within an infrastructure?

In this thesis, we proved that effectively supporting health data-sharing use cases hinges on the ability to adapt the infrastructure to promote a collaborative environment across healthcare institutions. To do that we need to ensure data control throughout the entire collaboration process. This is made possible by formalising exhaustive data-sharing policies that reflect the rules of the control each party needs. These need to be manageable and machine-understandable to be able to enforce and reason about these policies. This will promote policy-compliant workflow orchestration. The effectiveness of use case support also depends on the reliability and performance of the EPI framework to meet the use case requirements in terms of latency, overhead, and computing resources.

Moreover, integrating technologies that enhance programmability, security, and privacy is pivotal. Implementing BFC topologies to provision data encryption, access control, and audit trails strengthens the infrastructure's ability to manage sensitive health information securely. Additionally, leveraging machine learning and AI algorithms can optimize BFC placements and network setups, further enhancing the framework's capabilities in managing the trade-offs between security, privacy and network performance while running a use case across distributed environments.

Our research not only supports compliance with regulatory requirements but also promotes collaborative research, personalized medicine initiatives, and improved patient care outcomes through data-driven decision-making. Ultimately, a well-established infrastructure cultivated by these principles ensures that health data-sharing initiatives are sustainable, secure, and aligned with evolving healthcare needs. Our work serves to promote data sharing acceptance in healthcare institutions, and we demonstrated the EPI framework's practical benefits by deploying a PoC in collaboration with UMCU and St. Antonius Hospital.

9.1 Future Work

Our work could be further expanded in the following directions:

9.1.1 Deploy ML and NLP to manage policy complexity

In this thesis, we establish uniform formats and languages that can be readily implemented across diverse healthcare systems and institutions. We define clear standards for the ease of policy implementation and for fostering smoother data-sharing processes across the healthcare landscape. A work in progress is to manage the policy complexity that can get exponentially hard to formalise. This is dependent on the policy dynamicity, life cycle management, and cross-domain policy integration.

An interesting avenue could be to leverage Machine Learning (ML) algorithms and Natural Language Processing (NLP) techniques to interpret, enforce, and even suggest modifications to complex policies. Advanced NLP technologies, such as Transformers, unsupervised learning, and generative pre-training, now deployed in tools like ChatGPT, can play a significant role in this regard. By utilizing these cutting-edge technologies, healthcare organizations can automate the interpretation and management of policies, making the process more efficient and less prone to human error.

For instance, Transformers can be used to parse and understand complex policy documents, extracting key requirements and compliance metrics automatically. Unsupervised learning can help in identifying patterns and anomalies in policy application across different domains, suggesting areas for improvement or highlighting potential compliance risks. Generative pre-training models, like those used in ChatGPT, can assist in drafting policy documents, generating compliance reports, and even creating tailored policy recommendations based on specific organizational needs.

By automating policy management processes through these advanced AI and NLP technologies, healthcare organizations can streamline compliance efforts and adapt more effectively to evolving regulatory requirements and technological advancements. This approach not only enhances the accuracy and efficiency of policy enforcement but also frees up valuable human resources to focus on more strategic tasks, ultimately fostering a more responsive and resilient healthcare data-sharing ecosystem.

9.1.2 Refine the framework to improve adoption and deployment

Deploying the EPI framework in the actual healthcare landscape presents a challenge with numerous opportunities for exploration and refinement. One critical avenue for future work involves conducting real test-bed implementation studies to assess the framework's effectiveness, scalability, and practicality across diverse healthcare settings. Collaborating closely with healthcare institutions, research centres, and regulatory bodies will be essential to gain insights into the framework's usability, interoperability, and impact on healthcare delivery.

Understanding the factors influencing user acceptance and adoption of the EPI framework is another crucial aspect for future exploration. Researchers can delve into stakeholders' perceptions, attitudes, and concerns regarding data-sharing policies, security measures, and usability features to refine the framework according to user needs. Prioritizing user-centric design principles will be key to enhancing user satisfaction and engagement with the system.

Ensuring seamless interoperability and integration with existing healthcare systems and data repositories is paramount for widespread adoption. Future efforts should focus on developing robust interfaces, APIs, and data standards to facilitate data exchange and integration across heterogeneous healthcare environments.

Navigating complex regulatory landscapes and ensuring compliance with data protection laws, privacy regulations, and ethical standards are significant challenges. Future research should prioritize the development of comprehensive governance frameworks, audit mechanisms, and compliance monitoring tools to uphold regulatory requirements and mitigate legal risks.

Moreover, while the EPI framework provides the means to optimally deploy and provision workflows, policies, and BFC, we identify some weak spots that could affect adaptation. It is imperative to recognize that the security of the EPI topology heavily relies on the robustness of BFC implementations. Similarly, the effectiveness of the policy depends on the formalisation of considering all edge cases and pinpointing conflicts. Therefore, future efforts should prioritize the implementation and verification of state-of-the-art security functions within BFC configurations. This includes exploring and integrating advanced encryption techniques, strong authentication mechanisms, and stringent access controls to fortify data security and privacy protections.

Continuously enhancing security and privacy features within the EPI framework is essential to safeguard sensitive healthcare information against evolving threats. Research efforts should explore innovative security measures tailored to the unique demands of healthcare environments. This includes proactive measures such as regular security audits, vulnerability assessments, and updates to address emerging security and privacy risks effectively.

Capacity building and training programs for healthcare professionals and IT personnel will be crucial to ensure effective implementation and utilization of the EPI framework. Developing educational resources and hands-on training can empower stakeholders with the knowledge and skills needed to leverage the framework effectively. By addressing these key areas of future work, stakeholders can collaboratively work towards deploying and accepting the EPI framework as a foundational tool for enabling secure, transparent, and ethical data sharing in real-world healthcare environments.

On the other hand, on the road towards acceptance, it is essential to emphasize the importance of the growth of shared data, particularly for training purposes, as well as the need for identifying and filtering erroneous data and

mitigating potential biases. As healthcare institutions increasingly rely on data-driven decision-making, the volume and quality of shared data become critical factors in ensuring accurate and reliable outcomes. Therefore, robust mechanisms must be in place to validate, clean, and maintain the integrity of shared health data.

While data validation and filtering processes have been out of scope for this thesis, these mechanisms are crucial in identifying and rectifying erroneous data, which can otherwise lead to misleading conclusions and compromised patient care, subsequently hindering acceptance and deployment of healthcare applications.

Furthermore, the adoption and deployment of health data sharing are not solely contingent upon having the right technical framework. The success of such initiatives also depends on the development of viable business models and supportive government policies. The cost-benefit analysis of data-sharing initiatives must be thoroughly examined to ensure that all stakeholders, including healthcare providers, patients, and policymakers, see tangible value in participating. Sustainable business models that provide incentives for data sharing while ensuring affordability and access are crucial for widespread adoption.

In summary, the EPI framework's wide adoption hinges not only on its technical robustness but also on the broader context of data growth, quality assurance, business models, and regulatory support. By addressing these factors, stakeholders can collaboratively work towards a resilient and effective health data-sharing ecosystem that enhances healthcare delivery and patient outcomes.

Appendix A

Privacy Threats Attack Tree

A.1 Privacy threats: Data in Storage Stage

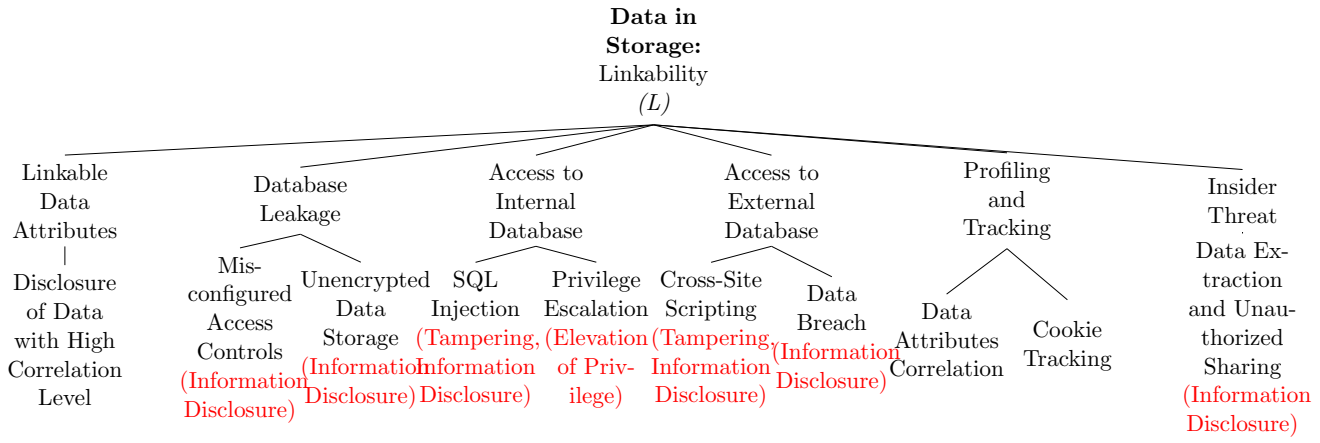


Figure A.1: Data in Storage: Linkability (L)

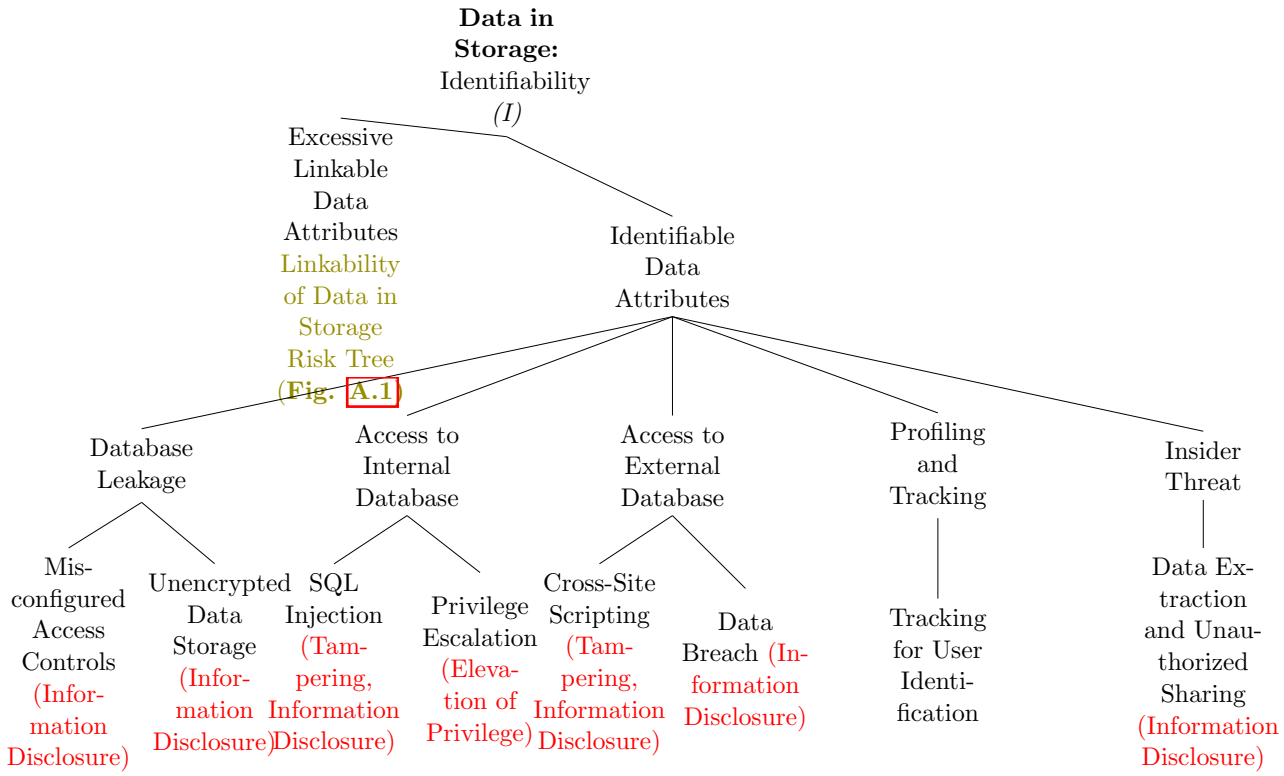


Figure A.2: Data in Storage: Identifiability (I)

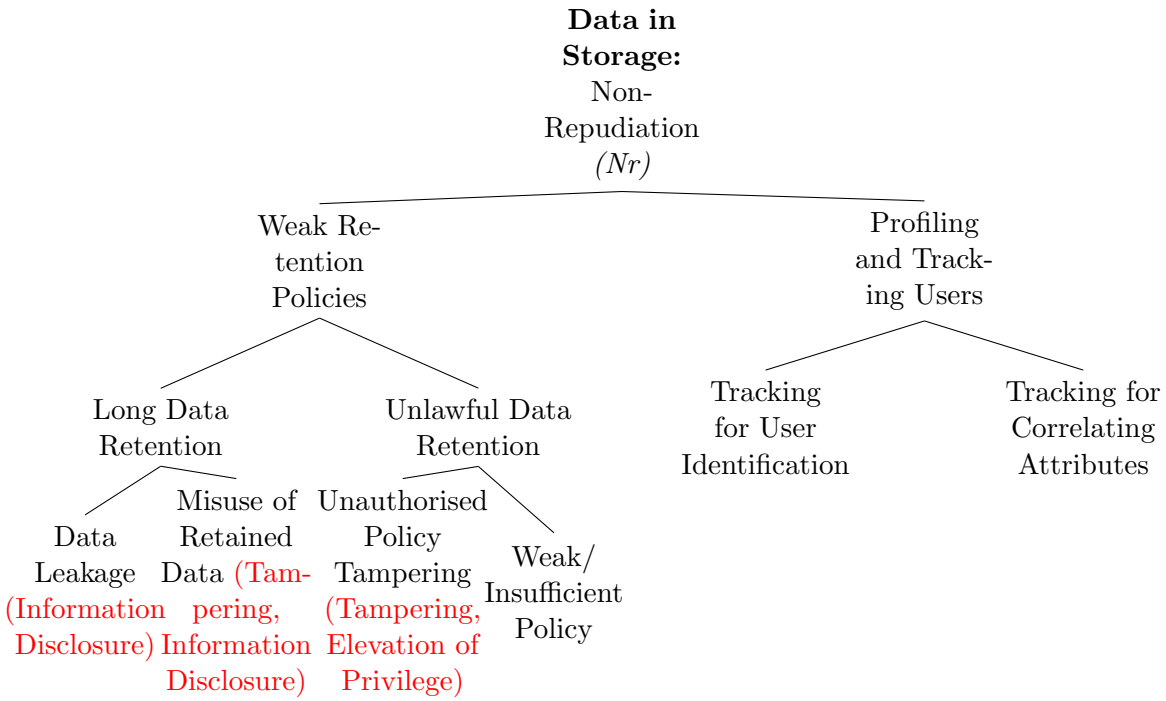


Figure A.3: Data in Storage: Non-Repudiation (Nr)

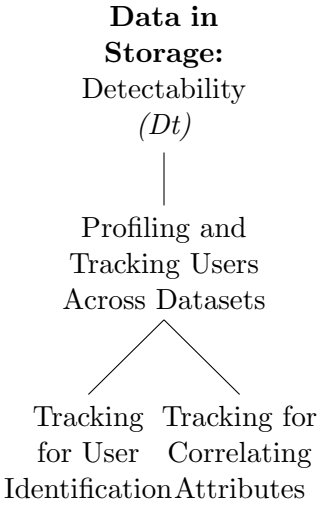


Figure A.4: Data in Storage: Detectability (Dt)

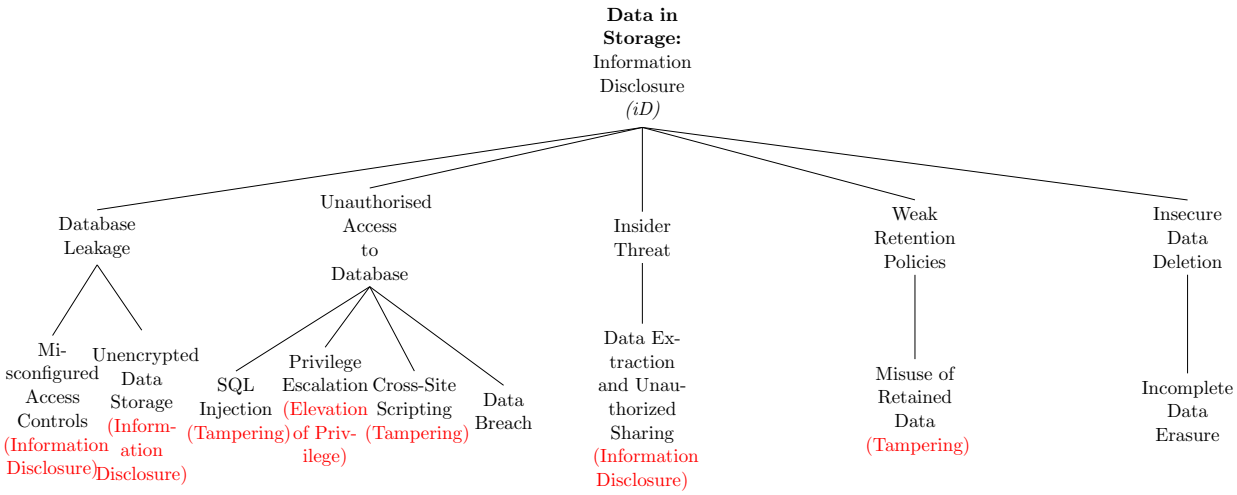


Figure A.5: Data in Storage: Information Disclosure (iD)

Figure A.6: Data in Storage: Indulgence (In)

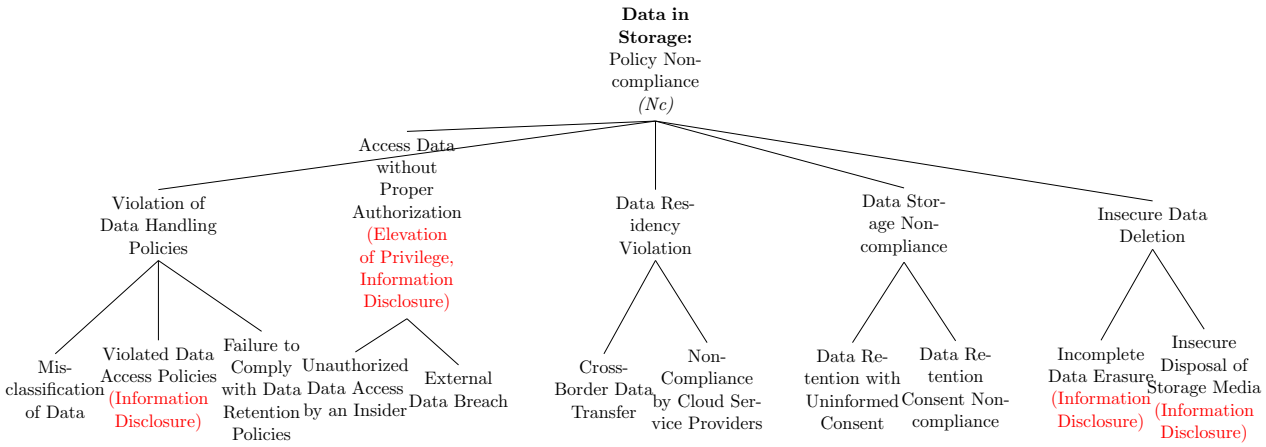


Figure A.7: Data in Storage: Policy Non-compliance (Nc)

A.2 Privacy threats: Data in Transit

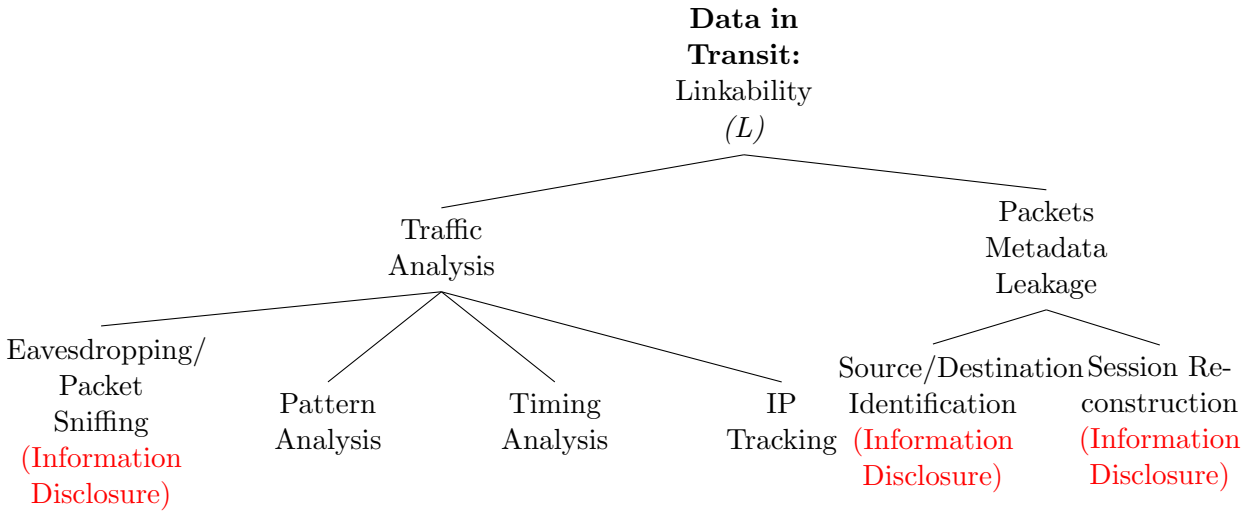


Figure A.8: Data in Transit: Linkability (L)

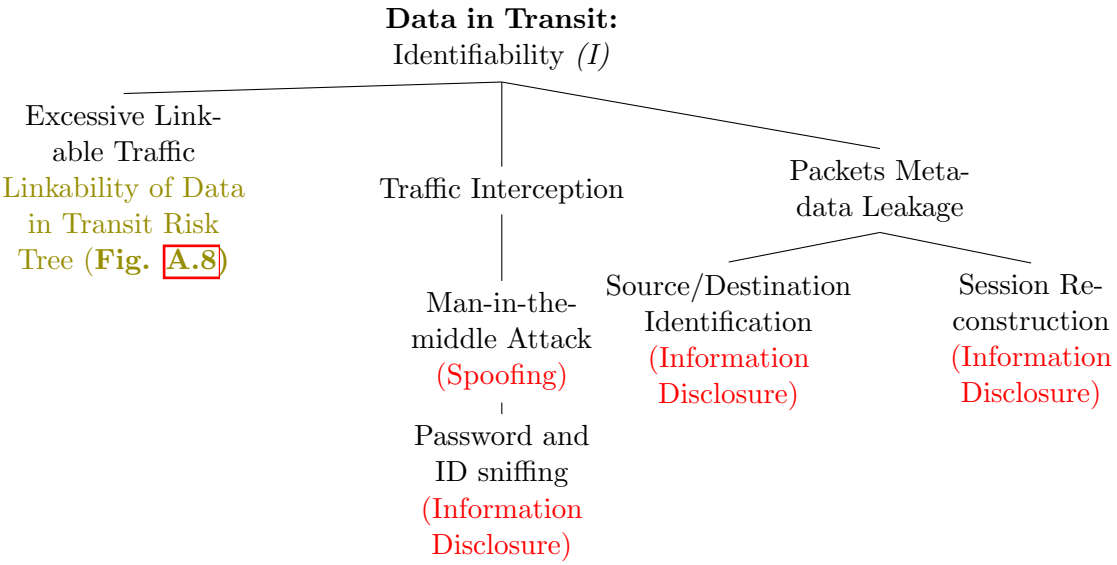


Figure A.9: Data in Transit: Identifiability (I)

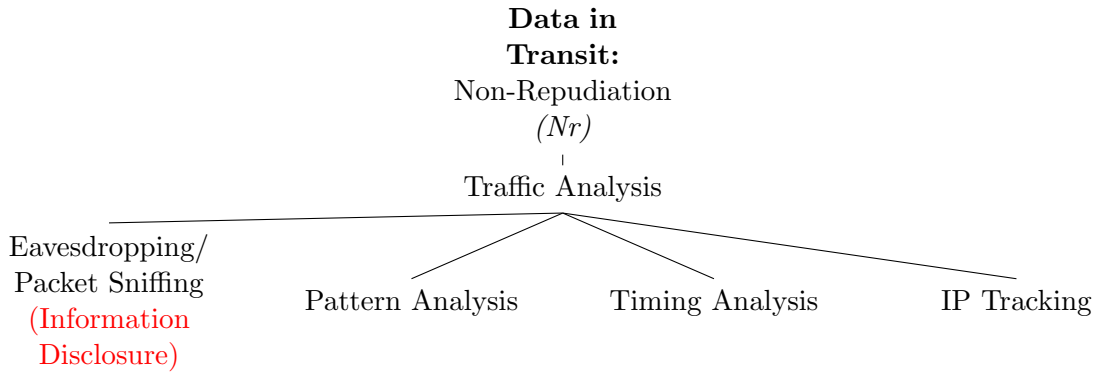


Figure A.10: Data in Transit: Non-Repudiation (Nr)

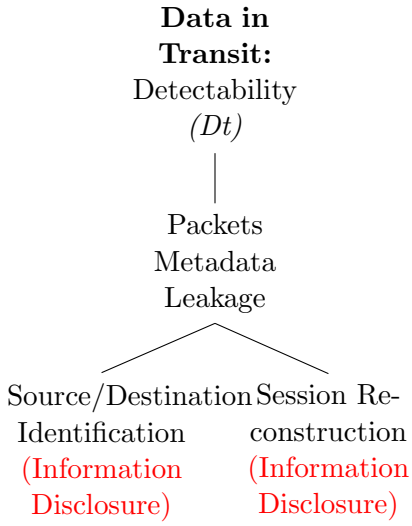


Figure A.11: Data in Transit: Detectability (Dt)

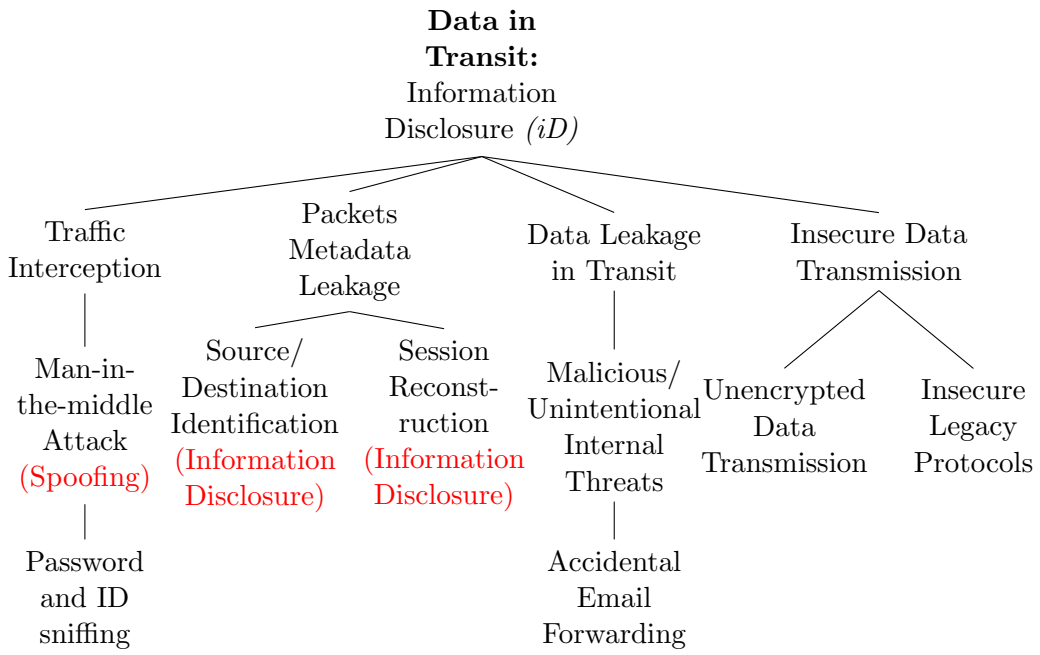


Figure A.12: Data in Transit: Information Disclosure (iD)

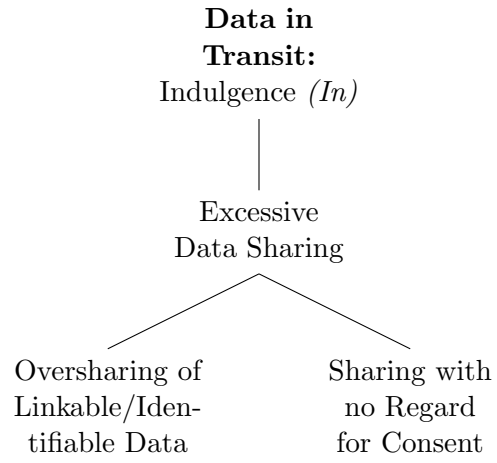


Figure A.13: Data in Transit: Indulgence (In)

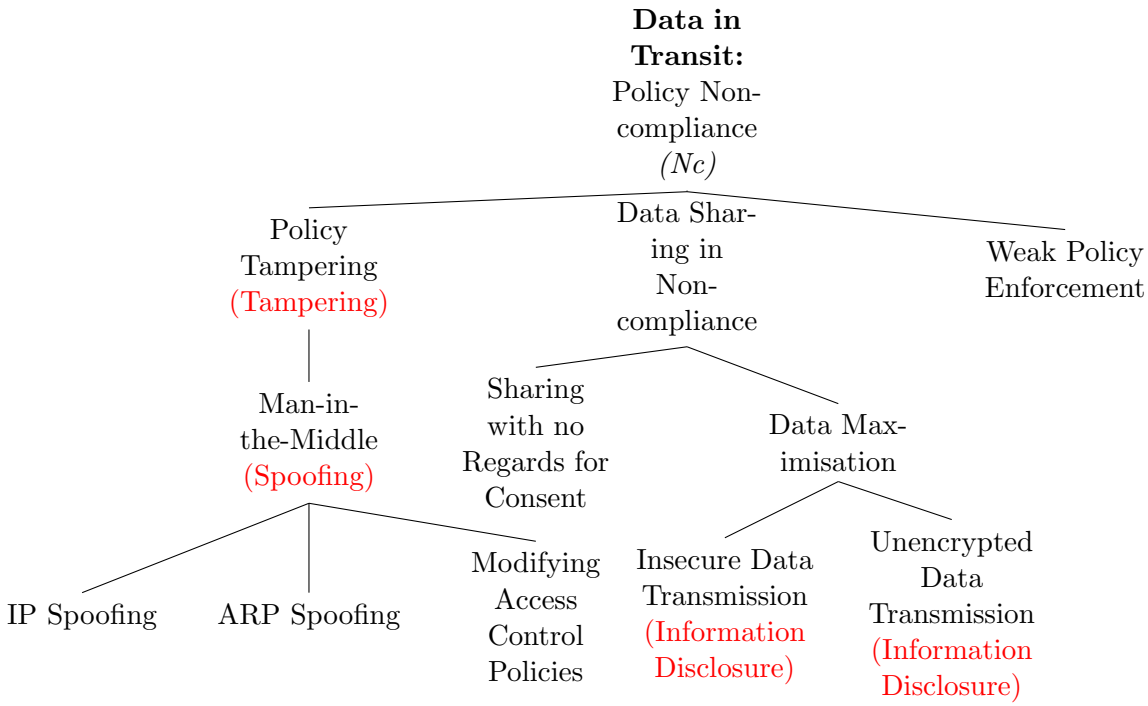


Figure A.14: Data in Transit: Policy Non-compliance (Nc)

A.3 Privacy Threats: Data in execution

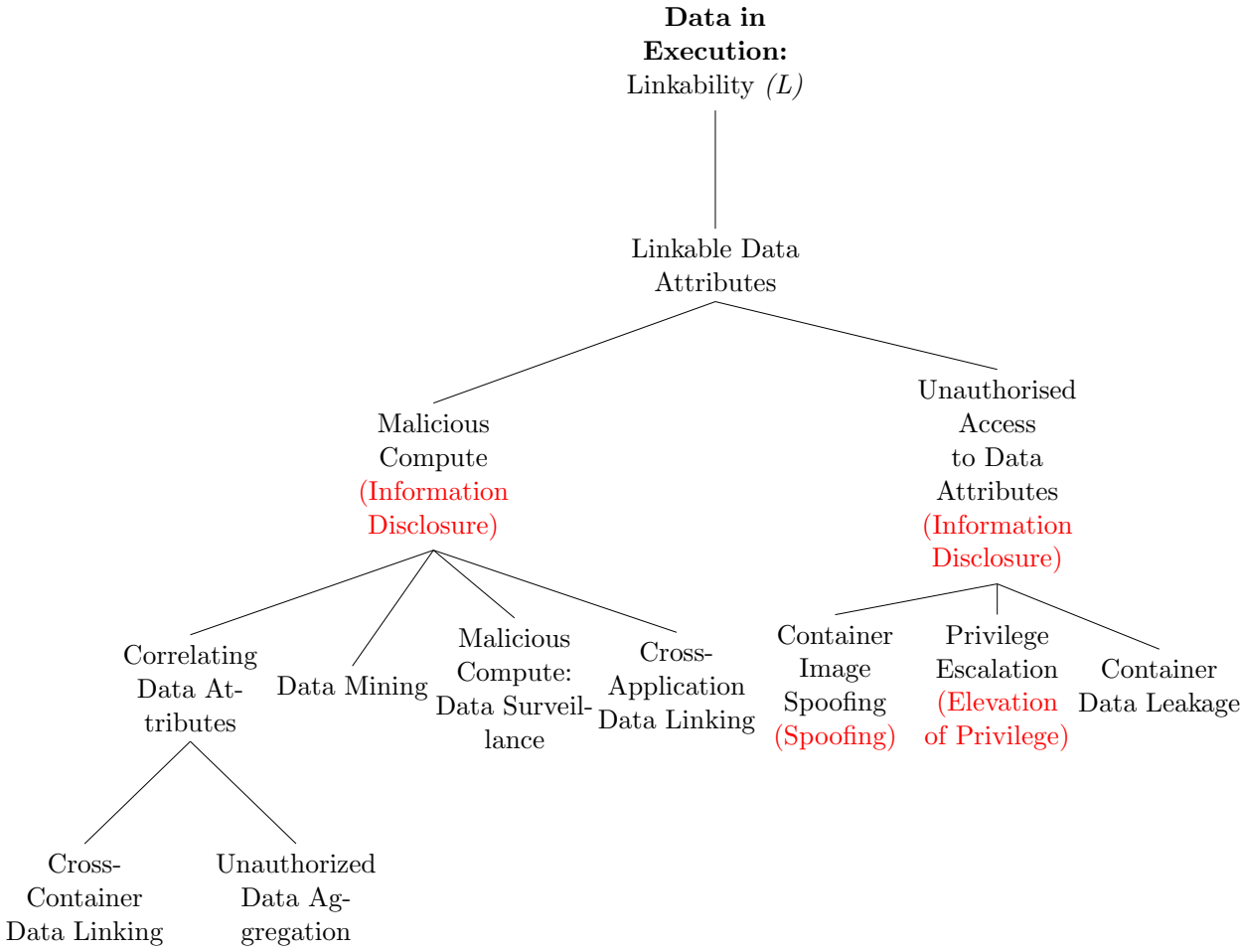


Figure A.15: Data in Execution: Linkability (L)

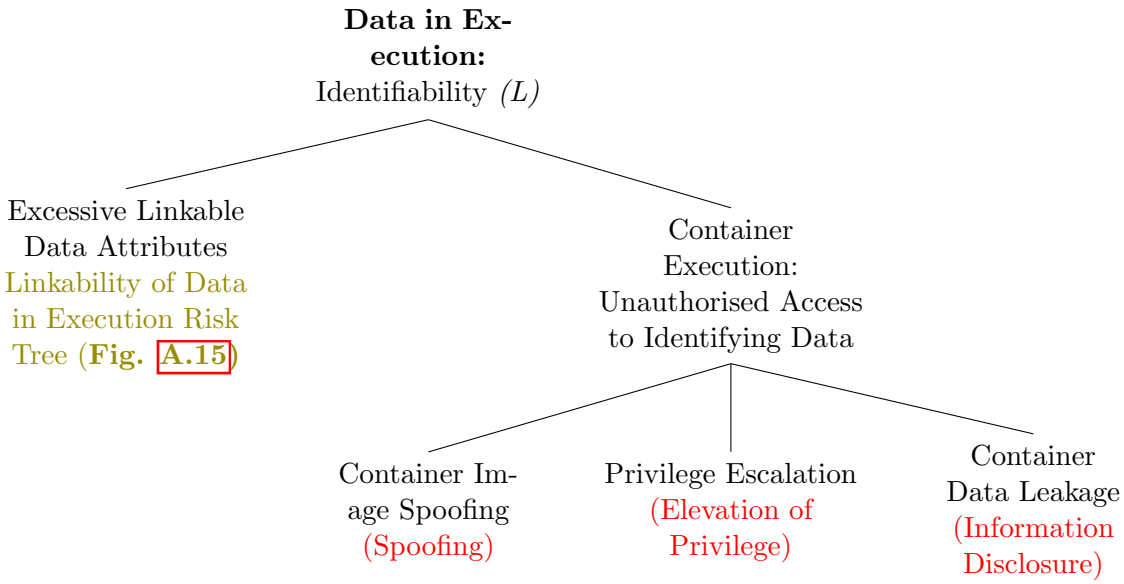


Figure A.16: Data in Execution: Identifiability (I)

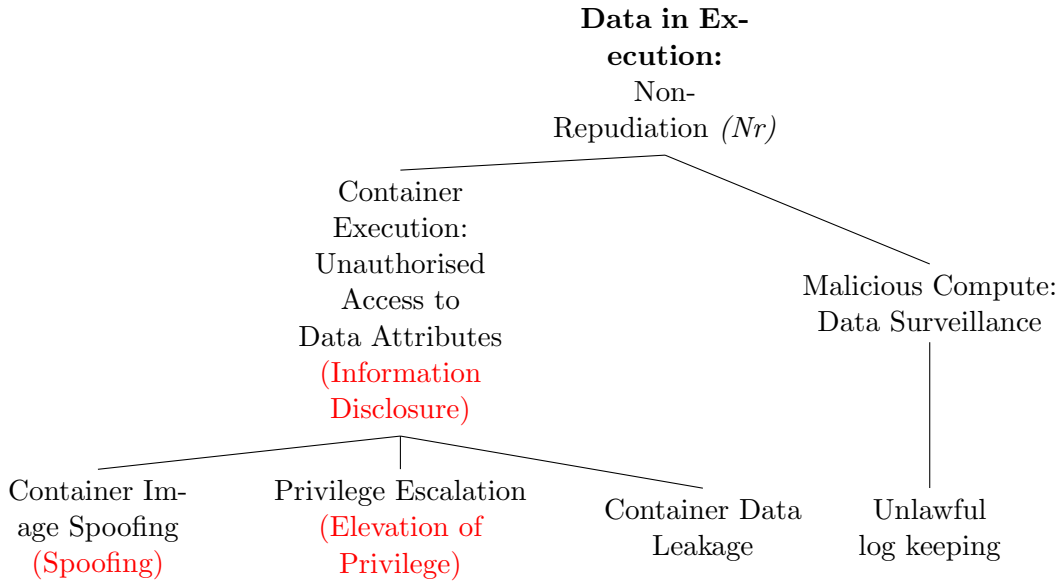


Figure A.17: Data in Execution: Non-Repudiation (Nr)

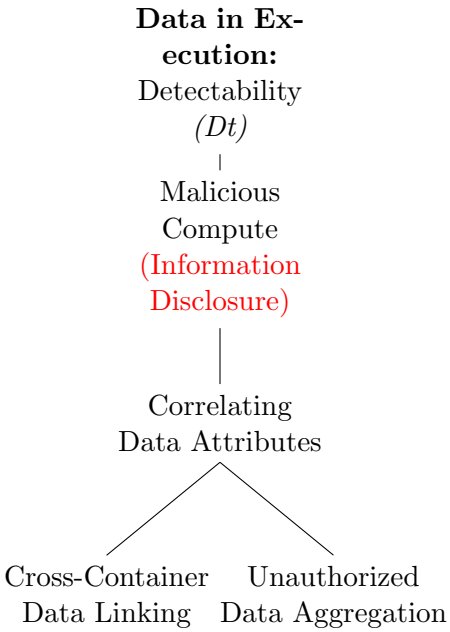


Figure A.18: Data in Execution: Detectability (Dt)

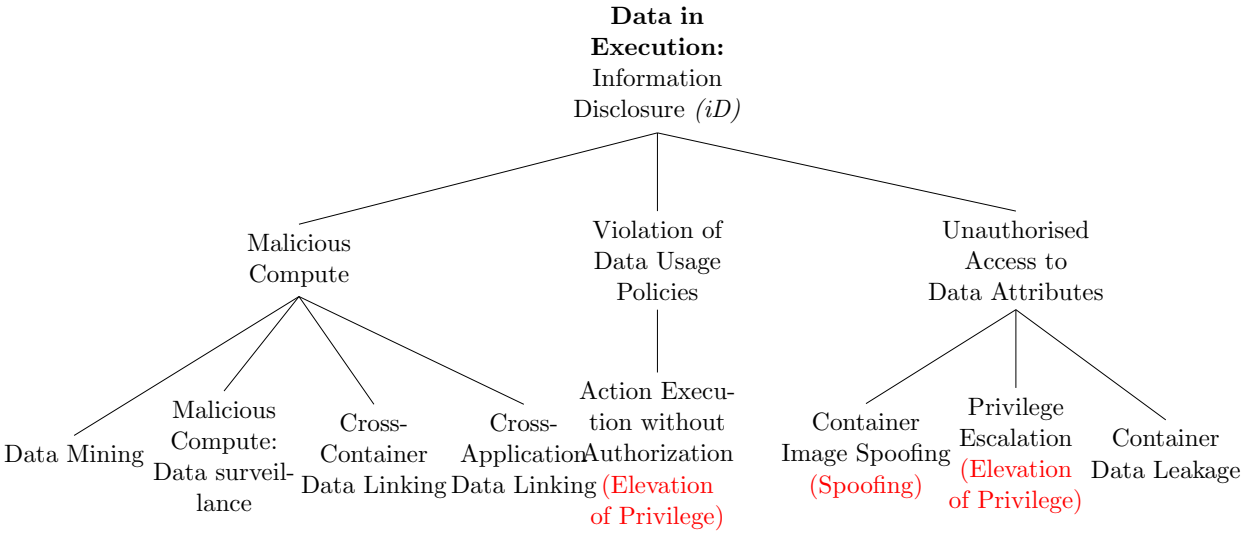


Figure A.19: Data in Execution: Information Disclosure (iD)

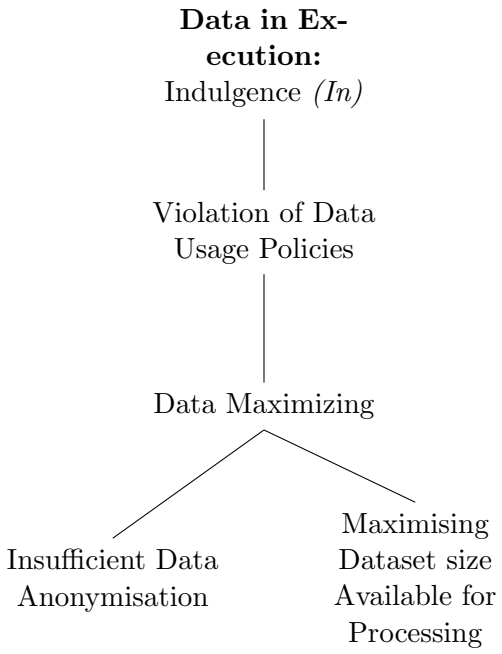


Figure A.20: Data in Execution: Indulgence (In)

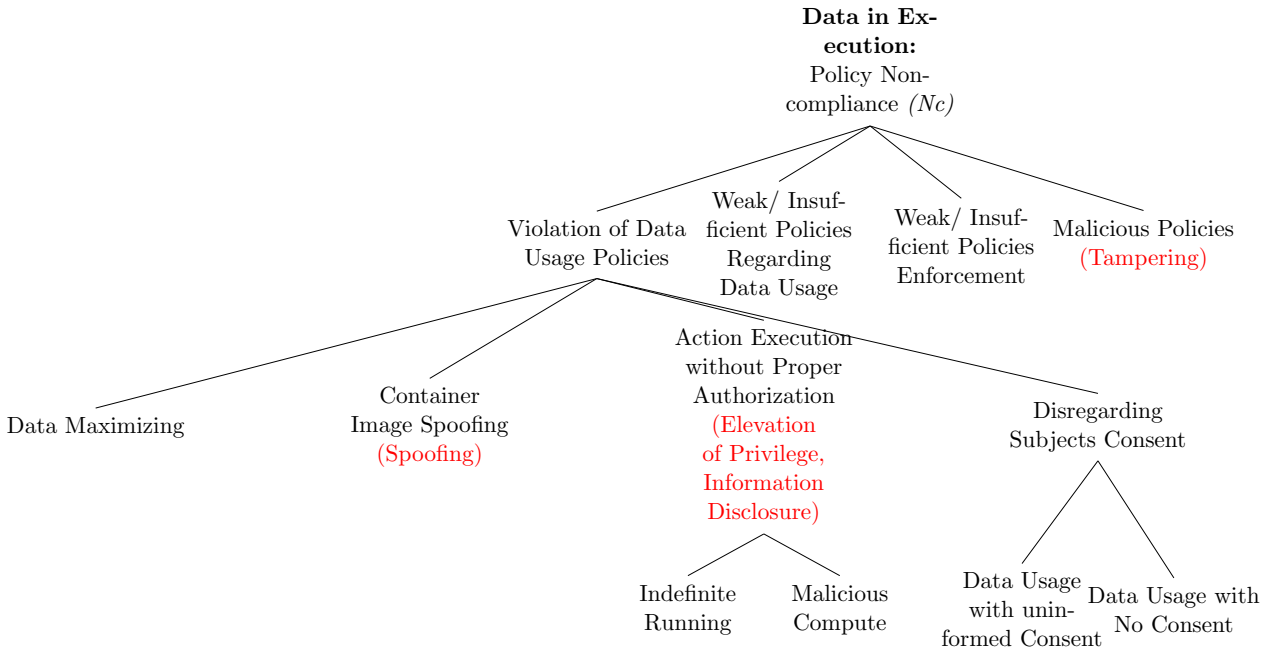


Figure A.21: Data in Execution: Policy Non-compliance (Nc)

Bibliography

[1] URL: https://www.etsi.org/technologies/nfv.

[2] https://delaat.net/epi/.

[3] httping(1) - linux man page. URL: https://linux.die.net/man/1/httping.

[4] Traffic redirection overview. URL: https://www.juniper.net/documentation/en_US/src4.13/topics/concept/redirect-server-overview.html.

[5] wg/wrk. URL: https://github.com/wg/wrk.

[6] Summary of the hipaa security rule, Jul 2013. URL: https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html.

[7] General Data Protection Regulation (GDPR): Recital 35 health data, Sep 2019. URL: https://gdpr-info.eu/recitals/no-35/.

[8] STRIDE/DREAD, the DREAD approach to threat assessment, 2020. URL: https://learn.microsoft.com/en-us/windows-hardware/drivers/driversecurity/threat-modeling-for-drivers.

[9] Nginx reverse proxy. Technical report, 9999. URL: https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/?_ga=2.225849692.737245468.1617007670-911961388.1613593493.

[10] Ahmed Abujoda and Panagiotis Papadimitriou. Midas: Middlebox discovery and selection for on-path flow processing. In *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–8, 2015. doi:10.1109/COMSNETS.2015.7098686.

[11] Sushant Agarwal, Simon Steyskal, Franjo Antunovic, and Sabrina Kirrane. Legislative compliance assessment: framework, model and gdpr instantiation. In *Privacy Technologies and Policy: 6th Annual Privacy Forum, APF 2018, Barcelona, Spain, June 13-14, 2018, Revised Selected Papers 6*, pages 131–149. Springer, 2018.

[12] Corinne G. Allaart, Björn Keyser, Henri Bal, and Aart van Halteren. Vertical split learning - an exploration of predictive performance in medical and other use cases. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022. doi:10.1109/IJCNN55064.2022.9891964.

[13] John A Ambrose and Rajat S Barua. The pathophysiology of cigarette smoking and cardiovascular disease: an update. *Journal of the American college of cardiology*, 43(10):1731–1737, 2004.

[14] N. A. Anoop and T. K. Parani. A real time efficient secure patient monitoring based on wireless body area networks. In *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, pages 1–5, Nov 2016. doi:10.1109/GET.2016.7916814.

[15] Joshua Baugh, Ute Bartels, James Leach, Blaise Jones, Brooklyn Chaney, Katherine E Warren, Jenavieve Kirkendall, Renee Doughman, Cynthia Hawkins, Lili Miles, et al. The international diffuse intrinsic pontine glioma registry: an infrastructure to accelerate collaborative research for an orphan disease. *Journal of neuro-oncology*, 132(2):323–331, 2017.

[16] John Bellessa, Evan Kroske, Reza Farivar, Mirko Montanari, Kevin Larson, and Roy H. Campbell. Netodessa: Dynamic policy enforcement in cloud networks. In *2011 IEEE 30th Symposium on Reliable Distributed Systems Workshops*, pages 57–61, 2011. doi:10.1109/SRDSW.2011.24.

[17] Henk J Blom, Gary M Shaw, Martin den Heijer, and Richard H Finnell. Neural tube defects and folate: case far from closed. *Nature Reviews Neuroscience*, 7(9):724–731, 2006.

[18] Giovanni Briganti, Marco Scutari, and Richard J McNally. A tutorial on bayesian networks for psychopathology researchers. *Psychological methods*, 2022.

[19] Koen Bruynseels, Filippo Santoni de Sio, and Jeroen van den Hoven. Digital twins in health care: Ethical implications of an emerging engineering paradigm. *Frontiers in Genetics*, 9, 2018. doi:10.3389/fgene.2018.00031.

[20] Ann Cavoukian et al. Privacy by design: The 7 foundational principles. *Information and privacy commissioner of Ontario, Canada*, 5:12, 2009.

[21] Ann Cavoukian, Angus Fisher, Scott Killen, and David A Hoffman. Remote home health care technologies: How to ensure privacy? build it in: Privacy by design. *Identity in the Information Society*, 3:363–378, 2010.

[22] Iker Ceballos, Vivek Sharma, Eduardo Mugica, Abhishek Singh, Alberto Roman, Praneeth Vepakomma, and Ramesh Raskar. Splitnn-driven vertical partitioning. *arXiv preprint arXiv:2008.04137*, 2020.

[23] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. Heuristic-guided reinforcement learning. *CoRR*, abs/2106.02757, 2021. URL: https://arxiv.org/abs/2106.02757, arXiv:2106.02757.

[24] Yuchung Cheng, Jerry Chu, Sivasankar Radhakrishnan, and Arvind Jain. TCP Fast Open. RFC 7413, December 2014. URL: https://rfc-editor.org/rfc/rfc7413.txt, doi:10.17487/RFC7413.

[25] Stuart Clayman, Elisa Maini, Alex Galis, Antonio Manzalini, and Nicola Mazzocca. The dynamic placement of virtual network functions. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–9, 2014. doi:10.1109/NOMS.2014.6838412.

[26] Council of the EU. General Data Protection Regulation. *Official Journal of the European Union*, 59, 2016.

[27] Lin Cui, Fung Po Tso, and Weijia Jia. Heterogeneous network policy enforcement in data centers. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 552–555, 2017. doi:10.23919/INM.2017.7987327.

[28] Richard Cziva and Dimitrios P. Pezaros. Container network functions: Bringing nfv to the network edge. *IEEE Communications Magazine*, 55(6):24–31, 2017. doi:10.1109/MCOM.2017.1601039.

[29] Ewa Deelman, Karan Vahi, Mats Rynge, Rajiv Mayani, Rafael Ferreira da Silva, George Papadimitriou, and Miron Livny. The evolution of the pegasus workflow management software. *Computing in Science Engineering*, 21(4):22–36, 2019. doi:10.1109/MCSE.2019.2919690.

[30] Vanessa Díaz-Zuccarini and Silvia Schievano. Biomedical imaging and computational modeling in cardiovascular disease: Patient-specific applications using numerical models. *Biomedical Imaging and Computational Modeling in Biomechanics*, pages 173–192, 2013.

[31] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE, 2018.

[32] Damian Eke, Ida E.J. Aasebø, Simisola Akintoye, William Knight, Alexandros Karakasidis, Ezequiel Mikulan, Paschal Ochang, George Ogoh, Robert Oostenveld, Andrea Pigorini, Bernd Carsten Stahl, Tonya White, and Lyuba Zehl. Pseudonymisation of neuroimages and data protection: Increasing access to data while retaining scientific utility. *Neuroimage: Reports*, 1(4):100053, 2021. URL: https://www.sciencedirect.com/science/article/pii/S2666956021000519, doi:https://doi.org/10.1016/j.ynirp.2021.100053.

[33] Glyn Elwyn, Dominick Frosch, Richard Thomson, Natalie Joseph-Williams, Amy Lloyd, Paul Kinnersley, Emma Cording, Dave Tomson, Carole Dodd, Stephen Rollnick, et al. Shared decision making: a model for clinical practice. *Journal of general internal medicine*, 27:1361–1367, 2012.

[34] Epictetus and Thomas W Higginson. *The Enchiridion*. Bobbs-Merrill, 1955.

[35] Christopher A. Esterhuyse, Tim Müller, L. Thomas van Binsbergen, and Adam S. Z. Belloum. Exploring the enforcement of private, dynamic policies on medical workflow execution. In *18th IEEE International Conference on e-Science, e-Science 2022, Salt Lake City, UT, USA, October 11-14, 2022*, pages 481–486. IEEE, 2022. doi:10.1109/eScience55777.2022.00086.

[36] Christopher A. Esterhuyse, Tim Müller, L. Thomas Van Binsbergen, and Adam S. Z. Belloum. Exploring the enforcement of private, dynamic policies on medical workflow execution. In *2022 IEEE 18th International Conference on e-Science (e-Science)*, pages 481–486, 2022. doi:10.1109/eScience55777.2022.00086.

[37] Christopher A. Esterhuyse, Tim Müller, L. Thomas Van Binsbergen, and Adam S. Z. Belloum. Exploring the enforcement of private, dynamic policies on medical workflow execution. In *2022 IEEE 18th International Conference on e-Science (e-Science)*, pages 481–486, 2022. doi:10.1109/eScience55777.2022.00086.

[38] European Commission. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance), 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[39] Kai Fan, Shangyang Wang, Yanhui Ren, Hui Li, and Yintang Yang. Medblock: Efficient and secure medical data sharing via blockchain. *Journal of Medical Systems*, 42(8):136, Jun 2018. doi:10.1007/s10916-018-0993-7.

[40] Wenjun Fan, Zhihui Du, David Fernández, and Xinning Hui. Dynamic hybrid honeypot system based transparent traffic redirection mechanism. In Sihan Qing, Eiji Okamoto, Kwangjo Kim, and Dongmei Liu, editors, *Information and Communications Security*, pages 311–319, Cham, 2016. Springer International Publishing.

[41] Alan Ford, Costin Raiciu, Mark J. Handley, Olivier Bonaventure, and Christoph Paasch. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 8684, march 2020. URL: https://rfc-editor.org/rfc/rfc8684.txt, doi:10.17487/RFC8684.

[42] David Hillel Gelernter. *Mirror World: Or the day software puts in the universe in a shoebox ...: How it will happen and what it will mean*. Oxford Univ. Press, 1991.

[43] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. *SIGCOMM Comput. Commun. Rev.*, 41(4):350–361, aug 2011. doi:10.1145/2043164.2018477.

[44] Aris Gkoulalas-Divanis and Grigorios Loukides. *Medical data privacy handbook*. Springer, 2015.

[45] M. Grieves. *Product Lifecycle Management:driving the next generation of Lean Thinking*. McGraw-Hill, 2006.

[46] Michael W. Grieves. Product lifecycle management: The new paradigm for enterprises. *International Journal of Product Development*, 2(1/2):71, 2005. doi:10.1504/ijpd.2005.006669.

[47] Kristen N. Griggs, Olya Ossipova, Christopher P. Kohlios, Alessandro N. Baccarini, Emily A. Howson, and Thaier Hayajneh. Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. *Journal of Medical Systems*, 42:1–7, 2018.

[48] Peter Grünwald. Beyond neyman-pearson. *arXiv preprint arXiv:2205.00901*, 2022.

[49] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015. doi:10.1109/MCOM.2015.7045396.

[50] Qijian He, Wei Yang, Bingren Chen, Yangyang Geng, and Liusheng Huang. Transnet: Training privacy-preserving neural network over transformed layer. *Proceedings of the VLDB Endowment*, 13(12):1849–1862, 2020.

[51] Lindsey M Hoffman, Sophie EM Veldhuijzen Van Zanten, Niclas Colditz, Joshua Baugh, Brooklyn Chaney, Marion Hoffmann, Adam Lane, Christine Fuller, Lili Miles, Cynthia Hawkins, et al. Clinical, radiologic, pathologic, and molecular characteristics of long-term survivors of diffuse intrinsic pontine glioma (dipg): a collaborative report from the international and european society for pediatric oncology dipg registries. *Journal of clinical oncology*, 36(19):1963, 2018.

[52] Steven R Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2), 2021.

[53] Tim Hulsen. Sharing is caring—data sharing initiatives in healthcare. *International Journal of Environmental Research and Public Health*, 17(9), 2020. URL: https://www.mdpi.com/1660-4601/17/9/3046, doi:10.3390/ijerph17093046.

[54] Renato Iannella and Serena Villata. Odrl information model 2.2. *W3C Recommendation*, 15, 2018.

[55] Abeer A. Z. Ibrahim, Fazirulhisyam Hashim, Nor K. Noordin, Aduwati Sali, Keivan Navaie, and Saber M. E. Fadul. Heuristic resource allocation algorithm for controller placement in multi-control 5g based on sdn/nfv architecture. *IEEE Access*, 9:2602–2617, 2021. doi:10.1109/ACCESS.2020.3047210.

[56] MHA Jansen, DG Van Vuurden, WP Vandertop, and GJL Kaspers. Diffuse intrinsic pontine gliomas: a systematic update on clinical trials and biology. *Cancer treatment reviews*, 38(1):27–35, 2012.

[57] Changqing Ji, Yu Li, Wenming Qiu, Uchechukwu Awada, and Keqiu Li. Big data processing in cloud computing environments. In *2012 12th international symposium on pervasive systems, algorithms and networks*, pages 17–23. IEEE, 2012.

[58] Simon Josefsson, Joachim Strombergson, and Nikos Mavrogiannopoulos. The Salsa20 Stream Cipher for Transport Layer Security. Internet-Draft draft-josefsson-salsa20-tls-04, Internet Engineering Task Force, November 2013. Work in Progress. URL: https://datatracker.ietf.org/doc/html/draft-josefsson-salsa20-tls-04.

[59] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[60] Maged N. Kamel Boulos and Peng Zhang. Digital twins: From personalised medicine to precision public health. *Journal of Personalized Medicine*, 11(8):745, 2021. doi:10.3390/jpm11080745.

[61] Jamila Alsayed Kassem, Adam Belloum, Tim Müller, and Paola Grosso. Utilisation profiles of bridging function chain for healthcare use cases. In *2022 IEEE 18th International Conference on e-Science (e-Science)*, pages 475–480, 2022. doi:10.1109/eScience55777.2022.00085.

[62] Jamila Alsayed Kassem, Adam Belloum, Tim Müller, and Paola Grosso. Utilisation profiles of bridging function chain for healthcare use cases. In *2022 IEEE 18th International Conference on e-Science (e-Science)*, pages 475–480, 2022. doi:10.1109/eScience55777.2022.00085.

[63] Jamila Alsayed Kassem, Cees De Laat, Arie Taal, and Paola Grosso. The epi framework: A dynamic data sharing framework for healthcare use cases. *IEEE Access*, 8:179909–179920, 2020. doi:10.1109/ACCESS.2020.3028051.

[64] Jamila Alsayed Kassem, Cees De Laat, Arie Taal, and Paola Grosso. The epi framework: A dynamic data sharing framework for healthcare use cases. *IEEE Access*, 8:179909–179920, 2020. doi:10.1109/ACCESS.2020.3028051.

[65] Jamila Alsayed Kassem, Cees De Laat, Arie Taal, and Paola Grosso. The epi framework: A dynamic data sharing framework for healthcare use cases. *IEEE Access*, 8:179909–179920, 2020. doi:10.1109/ACCESS.2020.3028051.

[66] Jamila Alsayed Kassem, Cees De Laat, Arie Taal, and Paola Grosso. The epi framework: A dynamic data sharing framework for healthcare use cases. *IEEE Access*, 8:179909–179920, 2020. doi:10.1109/ACCESS.2020.3028051.

[67] Jamila Alsayed Kassem, Onno Valkering, Adam Belloum, and Paola Grosso. Epi framework: Approach for traffic redirection through containerised network functions. In *2021 IEEE 17th International Conference on eScience (eScience)*, pages 80–89, 2021. doi:10.1109/eScience51609.2021.00018.

[68] Jamila Alsayed Kassem, Onno Valkering, Adam Belloum, and Paola Grosso. Epi framework: Approach for traffic redirection through containerised network functions. In *2021 IEEE 17th International Conference on eScience (eScience)*, pages 80–89, 2021. doi:10.1109/eScience51609.2021.00018.

[69] Jamila Alsayed Kassem, Li Zhong, Arie Taal, and Paola Grosso. Adaptive services function chain orchestration for digital health twin use cases: Heuristic-boosted q-learning approach, 2023. arXiv:2304.12853.

[70] Milen G Kebede, Giovanni Sileno, and Tom Van Engers. A critical reflection on odrl. In *AI Approaches to the Complexity of Legal Systems XI-XII: AICOL International Workshops 2018 and 2020: AICOL-XI@ JURIX 2018, AICOL-XII@ JURIX 2020, XAILA@ JURIX 2020, Revised Selected Papers XII*, pages 48–61. Springer, 2021.

[71] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[72] Elias Konidis, Panagiotis Kokkinos, and Emmanouel Varvarigos. Evaluating traffic redirection mechanisms for high availability servers. *2016 IEEE Globecom Workshops (GC Wkshps)*, 2016. doi:10.1109/glocomw.2016.7848898.

[73] Jaehoon Koo, Veena B. Mendiratta, Muntasir Raihan Rahman, and Anwar Walid. Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics, 2019. URL: https://arxiv.org/abs/1908.03242, doi:10.48550/ARXIV.1908.03242.

[74] Shilpa Krishnan, Catherine C Hay, Monique R Pappadis, Anne Deutsch, and Timothy A Reistetter. Stroke survivors' perspectives on post–acute rehabilitation options, goals, satisfaction, and transition to home. *Journal of Neurologic Physical Therapy*, 43(3):160–167, 2019.

[75] Romain Laborde, Michel Kamel, François Barrère, and Abdelmalek Benzekri. Implementation of a formal security policy refinement process in wbem architecture. *Journal of Network and Systems Management*, 15(2):241–266, 2007. doi:10.1007/s10922-007-9063-z.

[76] Marc Langheinrich. Privacy by design—principles of privacy-aware ubiquitous systems. In *International conference on ubiquitous computing*, pages 273–291. Springer, 2001.

[77] Peter Langhorne, Julie Bernhardt, and Gert Kwakkel. Stroke rehabilitation. *The Lancet*, 377(9778):1693–1702, 2011. URL: https://www.sciencedirect.com/science/article/pii/S0140673611603255, doi:https://doi.org/10.1016/S0140-6736(11)60325-5.

[78] Marcus D. Leech. SOCKS Protocol Version 5. RFC 1928, March 1996. URL: https://rfc-editor.org/rfc/rfc1928.txt, doi:10.17487/RFC1928.

[79] Marcus D. Leech. SOCKS Protocol Version 5. RFC 1928, March 1996. URL: https://www.rfc-editor.org/info/rfc1928, doi:10.17487/RFC1928.

[80] Ninghui Li. *Delegation Logic: A Logic-based Approach to Distrbuted Authorization*. PhD thesis, New York University, USA, 2000. URL: https://cs.nyu.edu/media/publications/li_ninghui.pdf.

[81] Gunasekaran Manogaran, R. Varatharajan, Daphne Lopez, Priyan Malarvizhi Kumar, Revathi Sundarasekar, and Chandu Thota. A new architecture of internet of things and big data ecosystem for secured smart healthcare monitoring and alerting system. *Future Generation Computer Systems*, 82:375 – 387, 2018. URL: http://www.sciencedirect.com/science/article/pii/S0167739X17305149, doi:https://doi.org/10.1016/j.future.2017.10.045.

[82] Maggie Mashaly. Connecting the twins: A review on digital twin technology & its networking requirements. *Procedia Computer Science*, 184:299–305, 2021. doi:10.1016/j.procs.2021.03.039.

[83] Mark I. McCarthy. Painting a new picture of personalised medicine for diabetes. *Diabetologia*, 60(5):793–799, 2017. doi:10.1007/s00125-017-4210-x.

[84] Arturo Moncada-Torres, Frank Martin, Melle Sieswerda, Johan van Soest, and Gijs Geleijnse. Vantage6: an open source privacy preserving federated learning infrastructure for secure insight exchange. In *AMIA Annual Symposium Proceedings*, pages 870–877, 2020.

[85] Yoav Nir and Adam Langley. ChaCha20 and Poly1305 for IETF Protocols. RFC 7539, May 2015. URL: https://www.rfc-editor.org/info/rfc7539, doi:10.17487/RFC7539.

[86] Guido Noordende, Silvia Olabarriaga, Matthijs Koot, and Laat M. Trusted data management for grid-based medical applications. 01 2011.

[87] Vladimir Olteanu and Dragos Niculescu. SOCKS Protocol Version 6. Internet-Draft draft-olteanu-intarea-socks-6-11, Internet Engineering Task Force, November 2020. Work in Progress. URL: https://datatracker.ietf.org/doc/html/draft-olteanu-intarea-socks-6-11.

[88] Linda M O'Keeffe, Gemma Taylor, Rachel R Huxley, Paul Mitchell, Mark Woodward, and Sanne AE Peters. Smoking as a risk factor for lung cancer in women and men: a systematic review and meta-analysis. *BMJ open*, 8(10):e021611, 2018.

[89] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[90] Vaishali Patel, Wesley Barker, and Erin Siminerio. Trends in consumer access and use of electronic health information. *Office of the National Coordinator for Health Information Technology: Washington DC.*, October 2015.

[91] Vishal Patel. A framework for secure and decentralized sharing of medical imaging data via blockchain consensus. *Health Informatics Journal*, 25(4):1398–1411, 2019. PMID: 29692204. doi:10.1177/1460458218769699.

[92] Charith Perera, Ciaran McCormick, Arosha K. Bandara, Blaine A. Price, and Bashar Nuseibeh. Privacy-by-design framework for assessing internet of things applications and platforms. In *Proceedings of the 6th International Conference on the Internet of Things*, IoT'16, page 83–92, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2991561.2991566.

[93] Wullianallur Raghupathi and Viju Raghupathi. Big data analytics in healthcare: promise and potential. *Health information science and systems*, 2(1):3, 2014.

[94] Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *arXiv preprint arXiv:2210.01948*, 2022.

[95] Erik Roelofs, Andre Dekker, Elisa Meldolesi, Ruud G. P. M. van Stiphout, Vincenzo Valentini, and Philippe Lambin. International data-sharing for radiotherapy research: An open-source based infrastructure for multicentric clinical data mining. *Radiotherapy and Oncology*, 110(2):370–374, 2 2014. doi:10.1016/j.radonc.2013.11.001.

[96] Kamran Sartipi, Krupa A. Kuriakose, and Weina Ma. An infrastructure for secure sharing of medical images between pacs and ehr systems. In *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '13, pages 245–259, Riverton, NJ, USA, 2013. IBM Corp. URL: http://dl.acm.org/citation.cfm?id=2555523.2555549.

[97] Farida Habib Semantha, Sami Azam, Kheng Cher Yeo, and Bharanidharan Shanmugam. A systematic literature review on privacy by design in the healthcare sector. *Electronics*, 9(3):452, 2020.

[98] S. Shakeri, V. Maccatrozzo, L. Veen, R. Bakhshi, L. Gommans, C. de Laat, and P. Grosso. Modeling and matching digital data marketplace policies. In *2019 15th International Conference on eScience (eScience)*, pages 570–577, 2019.

[99] OASIS Standard. extensible access control markup language (xacml) version 3.0. *A:(22 January 2013). URl: http://docs. oasis-open. org/xacml/3.0/xacml-3.0-core-spec-os-en. html*, 2013.

[100] Judith ter Schure and Peter Grünwald. Accumulation bias in meta-analysis: the need to consider time in error control. *F1000Research*, 8, 2019.

[101] Danan Thilakanathan, Shiping Chen, Surya Nepal, Rafael Calvo, and Leila Alem. A platform for secure monitoring and sharing of generic health data in the cloud. *Future Generation Computer Systems*, 35:102 – 113, 2014. Special Section: Integration of Cloud Computing and Body Sensor Networks; Guest Editors: Giancarlo Fortino and Mukaddim Pathan. URL: http://www.sciencedirect.com/science/article/pii/S0167739X13001908, doi:https://doi.org/10.1016/j.future.2013.09.011.

[102] Rosanne J. Turner, Femke Coenen, Femke Roelofs, Karin Hagoort, Aki Härmä, Peter D. Grünwald, Fleur P. Velders, and Floortje E. Scheepers. Information extraction from free text for aiding transdiagnostic psychiatry: constructing nlp pipelines tailored to clinicians' needs. *BMC psychiatry*, 22(1):407, 2022.

[103] Rosanne J. Turner and Peter D. Grünwald. Safe sequential testing and effect estimation in stratified count data. *arXiv preprint arXiv:2302.11401*, 2023.

[104] Rosanne J. Turner and Peter D. Grünwald. Exact anytime-valid confidence intervals for contingency tables and beyond. *Statistics & Probability Letters*, page 109835, 2023. URL: https://www.sciencedirect.com/science/article/pii/S0167715223000597, doi:https://doi.org/10.1016/j.spl.2023.109835.

[105] Frits W. Vaandrager, Bharat Garhewal, Jurriaan Rot, and Thorsten Wißmann. A new approach for active automata learning based on apartness. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, volume 13243 of *Lecture Notes in Computer Science*, pages 223–243. Springer, 2022. doi:10.1007/978-3-030-99524-9_12.

[106] Onno Valkering, Reginald Cushing, and A. Belloum. Brane: A framework for programmable orchestration of multi-site applications. pages 277–282, 09 2021. doi:10.1109/eScience51609.2021.00056.

[107] Onno Valkering, Reginald Cushing, and Adam Belloum. Brane: A framework for programmable orchestration of multi-site applications. In *17th IEEE International Conference on eScience, eScience 2021, Innsbruck, Austria, September 20-23, 2021*, pages 277–282. IEEE, 2021. doi:10.1109/eScience51609.2021.00056.

[108] L Thomas van Binsbergen, Milen G Kebede, Joshua Baugh, Tom Van Engers, and Dannis G van Vuurden. Dynamic generation of access control policies from social policies. *Procedia Computer Science*, 198:140–147, 2022.

[109] L. Thomas van Binsbergen, Lu-Chi Liu, Robert van Doesburg, and Tom van Engers. Eflint: A domain-specific language for executable norm specifications. In *Proceedings of the 19th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, GPCE 2020, page 124–136, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3425898.3426958.

[110] Lourens E. Veen, Sara Shakeri, and Paola Grosso. Mahiru: a federated, policy-driven data processing and exchange system, 2022. arXiv:2210.17155.

[111] Lourens E Veen, Sara Shakeri, and Paola Grosso. Mahiru: a federated, policy-driven data processing and exchange system. *arXiv preprint arXiv:2210.17155*, 2022.

[112] R. Venkateswaran. Virtual private networks. *IEEE Potentials*, 20(1):11–15, 2001. doi:10.1109/45.913204.

[113] Eric-Jan Wagenmakers. A practical solution to the pervasive problems of p values. *Psychonomic bulletin & review*, 14(5):779–804, 2007.

[114] Tom Jenno Wassing, Danny De Vleeschauwer, and Chrysa Papagianni. A machine learning approach for service function chain embedding in cloud datacenter networks. In *2021 IEEE 10th International Conference on Cloud Networking (CloudNet)*, pages 26–32, 2021. doi:10.1109/CloudNet53349.2021.9657124.

[115] Jun Wu, Zhifeng Zhang, Yu Hong, and Yonggang Wen. Cloud radio access network (c-ran): a primer. *IEEE Network*, 29(1):35–41, 2015. doi:10.1109/MNET.2015.7018201.

[116] Kim Wuyts and Wouter Joosen. Linddun privacy threat modeling: a tutorial. *CW Reports*, 2015.

[117] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani. Medshare: Trust-less medical data sharing among cloud service providers via blockchain. *IEEE Access*, 5:14757–14767, 2017. doi:10.1109/ACCESS.2017.2730843.

[118] Yi Yue, Bo Cheng, Meng Wang, Biyi Li, Xuan Liu, and Junliang Chen. Throughput optimization and delay guarantee vnf placement for mapping sfc requests in nfv-enabled networks. *IEEE Transactions on Network and Service Management*, 18(4):4247–4262, 2021. doi:10.1109/TNSM.2021.3087838.

[119] Lu Zhang, Arie Taal, Reginald Cushing, Cees Laat, and Paola Grosso. A risk-level assessment system based on the stride/dread model for digital data marketplaces. *International Journal of Information Security*, 21, 06 2022. doi:10.1007/s10207-021-00566-3.

[120] Qingchen Zhang, Laurence T. Yang, Zhikui Chen, and Peng Li. A survey on deep learning for big data. *Information Fusion*, 42:146–157, 2018. doi:10.1016/j.inffus.2017.10.006.

[121] Tianwei Zhang, Zecheng He, and Ruby B Lee. Privacy-preserving machine learning through data obfuscation. *arXiv preprint arXiv:1807.01860*, 2018.

List of symbols

DT: Digital Twin
DHT: Digital Health Twin
AI: Artificial Intelligence
EHR: Electronic Health Records
NVF: Network Function Virtualisation
NF: Network Function
EPI: Enabling Personalised Intervention
DPI: Deep Packet Inspection
SDI: Software Defined Infrastructure
CVA: Cerebral Vascular Accident
ECG: Electrocardiogram
BFC: Bridging Function Chain
BF: Bridging Function
SFC: Service Function Chain
DQL: Deep Q-Learning
RL: Reinforcement Learning
HDQL: Heuristic-based Deep Q-Learning
AMdEX: Amsterdam Data Exchange
DSA: Data Sharing Agreement
GDPR: General Data Protection Regulation
HIPAA: Health Insurance Portability and Accountability Act
SDM: Shared Decision Making
ODRL: Open Digital Rights Language
SIOPE: European Society for Paediatric Oncology
SAVI: Safe Anytime-Valid Inference
XACML: Access Control Markup Language
DIPG: Diffuse Intrinsic Pontine Glioma
MDP: Markov Decision Process
ML: Machine Learning
PPML: Privacy-Preserving Machine Learning
PoC: Proof of Concept
ECT: ElectroConvulsive Therapy
VPN: Virtual Private Network
PACS: Picture Archiving and Communication System
HL7: Health Level 7
BSN: Body Sensor Network
DICOM: Digital Imaging and Communications in Medicine
DSS: Data-Sharing Service
IoT: Internet of Things
PKI: Public key infrastructure
ISA: Information Sharing Agreement
IFC: Information Flow Control
TLS: Transport Layer Security
EPIF: EPI Framework
HTTP: Hypertext Transfer Protocol
VM: Virtual Machine
TCP: Transmission Control Protocol
UDP: User Datagram Protocol
RTT: Round-Trip Time
SDN: Software Defined Networking
NF: Network Function
CNF: Containerised Network Function

CNFV: Containerised Network Function Virtualisation
QoS: Quality of Service
HD: Hard Disk
NIC: Network Interface Card
SSL: Secure Sockets Layer
SSH: Secure Shell Protocol
NPoP: Network Points of Placement
HI: Healthcare Institution
NAT: Network Address Translation
PbD: Privacy by Design
CRF: Cardiovascular Research Foundation
SSIM: Structural Similarity Index
PSNR: Peak Signal-to-Noise Ratio
SFTP: SSH File Transfer Protocol
NFS: Network File System
SMPC: Secure Multi-Party Computation
GPCE: General-Purpose Computing Environments

Summary

This thesis tackles the pressing challenge of secure and efficient health data sharing, proposing the Enabling Personalized Intervention (EPI) framework designed to facilitate the data collaboration process across healthcare institutions. This thesis introduces a framework that provides seamless, policy-compliant sharing capabilities within a multi-party environment. The EPI framework is structured to support a flexible yet highly secure infrastructure, accommodating the diverse needs of healthcare providers who must share data across institution boundaries. The architecture is composed of three interdependent components: the Policy Reasoner, the Infrastructure Orchestrator, and the Workflow Orchestrator. Each component is designed to handle specific aspects of the data-sharing process, from enforcing policies to managing resource allocation and orchestrating workflows.

The Policy Reasoner interprets and applies rules for data sharing, ensuring that access and usage adhere strictly to both institutional and regulatory policies. The Infrastructure Orchestrator manages computational resources dynamically, adapting to varying demands in data processing and network requirements, and the Workflow Orchestrator coordinates the secure flow of data between institutions, aligning workflow operations with set security and privacy constraints. Together, these components create a cohesive system capable of securely sharing health data while respecting stringent compliance standards.

At the core of the EPI framework is its policy-driven approach, which integrates rules around privacy and access control directly into the data-sharing infrastructure. This approach allows healthcare institutions to share data in ways that comply with both internal policies and broader regulatory guidelines, maintaining high standards of patient confidentiality and security. The framework also incorporates dynamic, adaptive resource allocation using Kubernetes-based containerization, allowing it to efficiently scale resources up or down depending on the data-sharing requirements. This adaptability enhances performance and supports a diverse range of data types and workflows, making the framework suitable for both complex, high-throughput tasks and simpler data exchanges.

Additionally, the EPI framework introduces a proxy-based traffic management solution to optimize data flow across healthcare systems.

This thesis explores automated Service Function Chain (SFC) provisioning, where the thesis deploys heuristic and AI-driven tools to dynamically assign resources based on workload requirements. The Heuristic-boosted Deep Q-Learning (HDQL) tool efficiently matches resources to data-sharing needs, reducing latency and maximizing CPU utilization.

The EPI framework is proven to improve privacy according to the LINDINN privacy model, which evaluates privacy risks and directs the application of policies that mitigate potential vulnerabilities. Experimental evaluations highlight the effectiveness of the EPI framework's privacy-by-design approach, showcasing its potential for privately sharing sensitive medical information between healthcare providers.

In summary, this thesis presents a comprehensive framework for secure and adaptable health data sharing, addressing the unique challenges of privacy, compliance, and efficiency in the healthcare sector. The EPI framework's combination of policy-driven controls, adaptive infrastructure management, and privacy risk mitigation provides a strong foundation for secure, compliant data sharing across institutions. With its capacity to scale resources and protect sensitive information, this framework represents a significant step toward enabling personalized medicine and advanced healthcare research in a collaborative, data-rich environment.

Samenvatting

Deze dissertatie richt zich op de dringende uitdaging van veilige en efficiënte gezondheidsdata uitwisseling en stelt het Enabling Personalized Intervention (EPI)-framework voor, ontworpen om het proces van datacollaboratie tussen zorginstellingen te faciliteren. Deze thesis introduceert een framework dat naadloze, beleidsconforme mogelijkheden biedt voor gegevensuitwisseling in een multi partijomgeving. Het EPI-framework is opgebouwd om een flexibele, maar zeer veilige infrastructuur te ondersteunen, die tegemoetkomt aan de uiteenlopende behoeften van zorgverleners die gegevens over institutionele grenzen heen moeten delen. De architectuur bestaat uit drie onderling afhankelijke componenten: de Policy Reasoner, de Infrastructure Orchestrator en de Workflow Orchestrator. Elke component is ontworpen om specifieke aspecten van het data uitwisselingsproces af te handelen, van het afdwingen van beleid tot het beheren van resourceallocatie en het coördineren van workflows.

De Policy Reasoner interpreteert en past regels toe voor gegevensdeling, waarbij wordt gegarandeerd dat toegang en gebruik strikt voldoen aan zowel institutionele als wettelijke beleidsregels. De Infrastructure Orchestrator beheert de computationele resources dynamisch, waarbij hij zich aanpast aan wisselende eisen op het gebied van gegevensverwerking en netwerkbehoeften. De Workflow Orchestrator coördineert de veilige stroom van gegevens tussen instellingen en stemt de workflow-operaties af op vastgestelde beveiligings- en privacybeperkingen. Samen vormen deze componenten een samenhangend systeem dat in staat is gezondheidsgegevens veilig te delen, met inachtneming van strikte nalevingsnormen.

De kern van het EPI-framework wordt gevormd door een beleidsgestuurde benadering, die privacy- en toegangsregels direct integreert in de data uitwisselingsinfrastructuur. Deze aanpak stelt zorginstellingen in staat om gegevens te delen op een manier die voldoet aan zowel interne beleidsregels als bredere wettelijke richtlijnen, waarbij hoge normen voor patiëntvertrouwelijkheid en veiligheid worden gehandhaafd. Het framework omvat ook dynamische, adaptieve resourceal-

locatie door gebruik te maken van op Kubernetes gebaseerde containerisatie, waardoor het efficiënt resources kan opschalen of verkleinen, afhankelijk van de vereisten voor gegevensuitwisseling. Deze aanpasbaarheid verbetert de prestaties en ondersteunt een divers scala aan gegevenstypen en workflows, waardoor het framework geschikt is voor zowel complexe, hoge-doorvoer taken als eenvoudigere gegevensuitwisselingen. Bovendien introduceert het EPI-framework een proxy-gebaseerde verkeersbeheersoplossing om de gegevensstroom tussen zorgsystemen te optimaliseren.

Deze thesis verkent geautomatiseerde Service Function Chain (SFC) voorzieningen, waarbij heuristische en AI-gestuurde tools worden ingezet om dynamisch resources toe te wijzen op basis van de werkbelastingvereisten. De Heuristic-boosted Deep Q-Learning (HDQL) tool koppelt resources efficiënt aan de behoeften van gegevensuitwisseling, waardoor latentie wordt verminderd en CPU-gebruik wordt gemaximaliseerd.

Het EPI-framework heeft aangetoond privacy te verbeteren volgens het LINDI-NN privacymodel, dat privacyrisico's beoordeelt en de toepassing van beleid stuurt om potentiële kwetsbaarheden te beperken. Experimentele evaluaties onderstrepen de effectiviteit van de privacy-by-design aanpak van het EPI-framework en tonen het potentieel aan om gevoelige medische informatie privé te delen tussen zorgverleners.

Samenvattend biedt deze thesis een uitgebreid framework voor veilige en aanpasbare gezondheidsdata uitwisseling, waarmee wordt ingespeeld op de unieke uitdagingen op het gebied van privacy, naleving en efficiëntie in de zorgsector. De combinatie van beleidsgestuurde controles, adaptief infrastructuurbeheer en mitigatie van privacyrisico's in het EPI-framework biedt een solide basis voor veilige, nalevingsgerichte gegevensuitwisseling tussen instellingen. Met de capaciteit om resources op te schalen en gevoelige informatie te beschermen, vertegenwoordigt dit framework een belangrijke stap in de richting van gepersonaliseerde geneeskunde en geavanceerd gezondheidsonderzoek in een collaboratieve, datarijke omgeving.

Publications

- J. A. Kassem, C. De Laat, A. Taal and P. Grosso, "The EPI Framework: A Dynamic Data Sharing Framework for Healthcare Use Cases," in IEEE Access, vol. 8, pp. 179909-179920, 2020, doi: 10.1109/ACCESS.2020.3028051.

- J. A. Kassem, O. Valkering, A. Belloum and P. Grosso, "EPI Framework: Approach for Traffic Redirection Through Containerised Network Functions," 2021 IEEE 17th International Conference on eScience (eScience), Innsbruck, Austria, 2021, pp. 80-89, doi: 10.1109/eScience51609.2021.00018.

- J. A. Kassem, A. Belloum, T. Müller and P. Grosso, "Utilisation Profiles of Bridging Function Chain for Healthcare Use Cases," 2022 IEEE 18th International Conference on e-Science (e-Science), Salt Lake City, UT, USA, 2022, pp. 475-480, doi: 10.1109/eScience55777.2022.00085.

- J. A. Kassem, L. Zhong, A. Taal and P. Grosso, "Adaptive Services Function Chain Orchestration For Digital Health Twin Use Cases: Heuristic-boosted Q-Learning Approach," 2023 IEEE 9th International Conference on Network Softwarization (NetSoft), Madrid, Spain, 2023, pp. 187-191, doi: 10.1109/NetSoft57336.2023.10175506.

- J. A. Kassem, C. Allaart, S. Amiri, M. Kebede, T. Müller, R. Turner, A. Belloum, L. T. v. Binsbergen, P. Grunwald, A. v. Halteren, P. Grosso, C. d. Laat, and S. Klous. Building a Digital Health Twin for Personalized Intervention: The EPI Project. In Commit2Data. Open Access Series in Informatics (OASIcs), Volume 124, pp. 2:1-2:18, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2024) https://doi.org/10.4230/OASIcs.Commit2Data.2

- J. A. Kassem, T. Müller, C. A. Esterhuyse, M. G. Kebede, C. de Laat, A. Osseyran and P. Grosso, The EPI framework: A data privacy by design framework to support healthcare use cases, Future Generation Computer

Systems, 2024, 107550, ISSN 0167-739X, https://doi.org/10.1016/j.future.2024.107550.

Acknowledgments

I am deeply grateful to my supervisor, Paola, for always making time for me. Learning from you over the years has profoundly shaped me as a researcher. You taught me the importance of balancing attention to detail with a broader perspective, and I am better for it. I am also very grateful to Cees for his invaluable guidance and feedback. Whenever I felt stuck in my research, you were the first person I turned to, always ready with new ideas and interesting avenues to explore. I am truly thankful to Anwar, whose constant support kept me grounded; your words of encouragement lifted me both in my work and on a personal level. Axel, even though our time working together was brief, it was an honour to collaborate with you.

My heartfelt thanks go out to my colleagues, Onno, Adam, Tim, Chris, and Milen. Your contributions to this thesis were invaluable, and I'll always fondly remember our Friday meetings that we never managed to start on time.

To my paranymphs, Sara and Lu, thank you for the wonderful times we shared, both inside and outside the office. Ruyue and Zeshun, thank you for the laughs and the engaging discussions. I look forward to many more talks in the future. A special thanks to my MNS colleagues—Zhiming, Chrysa, Florian, Cyril, Floris-Jan, Henk, Leonardo, Na, Saba, Yuandou, Ralph, Reggie, Gabriel, Joseph, Spiros, and Misha. Mondays were a little less like Mondays with the energy of your lively discussions.

I am also grateful for the wider SNE cluster's colleagues and everyone in the EPI team, particularly Thomas, Sander, Francesco, Tom, Giovanni, Ana, Yuri, Leon, Guido, Lu-Chi, Xin, Yixian, Kyrian, Daphnee, Marco, Mostafa, and Grace. Thank you all for your camaraderie and support throughout this journey.

To my family—my mum, dad, and wonderful siblings—thank you for your unwavering love, support, and belief in me. Mum, you were always there with a listening ear whenever I felt overwhelmed, and I'm forever grateful for your steady presence. Dad, your support has been the foundation of everything I've achieved; I truly couldn't have come this far without you. To my brother, Ali,

thank you for listening with such genuine interest as I explained my research, even if you never questioned my logic—I appreciate that more than you know. Alaa, thank you for helping me pack and prepare for my move to the Netherlands; your encouragement and love made all the difference.

A special thanks to my little brother, Sajed, who has been a constant source of motivation. Knowing I am his role model inspires me to work harder every day and to be worthy of that honor. Finally, to my partner, Bassel—thank you for being by my side, and always motivating me to get things done, and helping me with the cover design. Your support means the world to me, and I'm grateful to share this accomplishment with you.